STC12C5410AD 系列单片机器件手册

STC12C2052AD 系列单片机器件手册

--- 1 个时钟 / 机器周期 8051
--- 无法解密
--- 低功耗,超低价
--- 高速，高可靠
--- 强抗静电，强抗干扰

STC12C5412, STC12C5412AD
STC12C5410, STC12C5410AD
STC12C5408, STC12C5408AD
STC12C5406, STC12C5406AD
STC12C5404, STC12C5404AD
STC12C5402, STC12C5402AD

STC12C5052, STC12C5052AD
STC12C4052, STC12C4052AD
STC12C2052, STC12C2052AD
STC12C1052, STC12C1052AD
STC12C0552, STC12C0552AD

宏晶科技
www.MCU-Memory.com
Update date: 2008-8-15

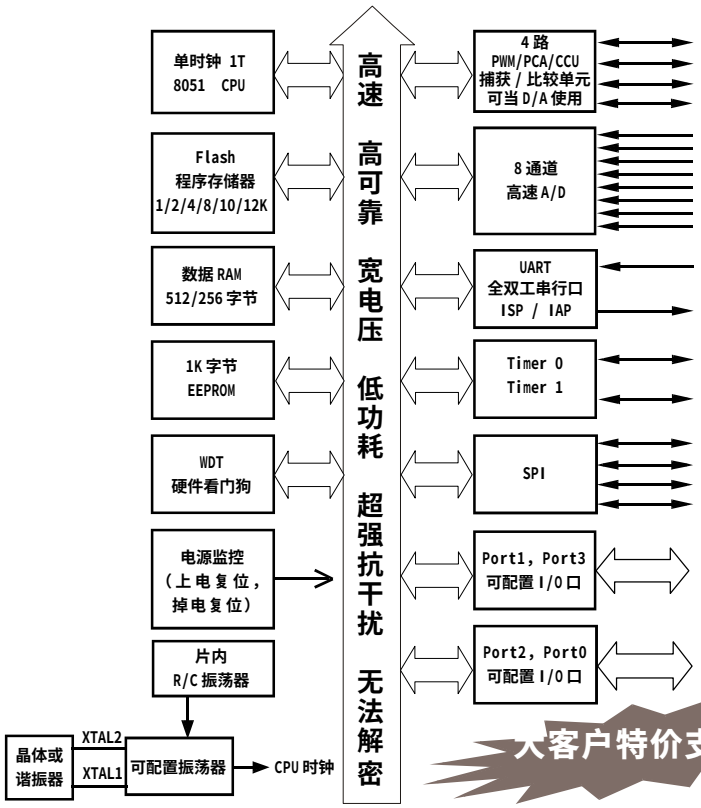
领导业界革命　覆盖市场需求

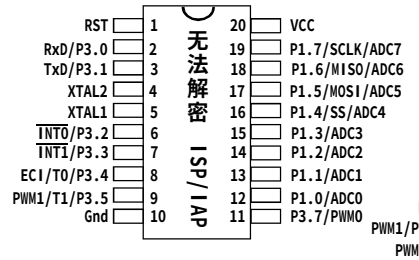
STC 12C5410AD 系列 1T 8051 单片机

——1 个时钟 / 机器周期，高速、高可靠，4 路 PWM，8 路高速 A/D 转换

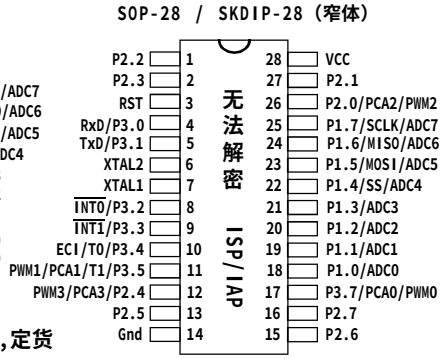
宏晶科技是新一代增强型 8051 单片机标准的制定者和领导厂商,致力于提供满足中国市场需求的世界级高性能单片机技术,在业内处于领先地位,销售网络覆盖全国。在高品质的基础上,以极低的价格和完善的服务赢得了客户的长期信赖。在广受欢迎的 STC89C51 全系列单片机的基础上，现全力推出“1 个时钟 / 机器周期”的单片机,全面提升 8051 单片机性能。欢迎海内外厂家前来洽谈合作!新客户请直接联系深圳,以获得更好的技术支持与服务。

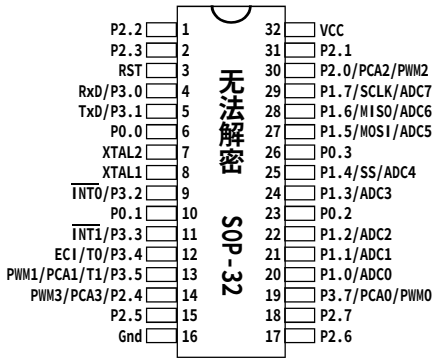
强烈推荐选择 SOP-20/28/32 贴片封装,传统的插件 DIP 封装稳定供货





超小封装 TSSOP-20,6.4mm X 6.4mm,定货





大客户特价支持

★如选 32-Pin，推荐选 SOP-32，LQFP-32

★如果 I/O 口不够用，可以用 2 到 3 根普通 I/O 口线外接 74HC164/165/595（均可级联）来扩展 I/O 口,还可用 A/D 做按键扫描来节省 I/O 口

STC12C5410/STC12C2052 系列主要性能：

- ●高速：1 个时钟 / 机器周期，增强型 8051 内核，速度比普通 8051 快 8～12 倍
- ●宽电压：5.5～3.5V，2.2～3.8V（STC12LE5410AD 系列）
- ●低功耗设计：空闲模式，掉电模式（可由外部中断唤醒）
- ●工作频率：0～35MHz，相当于普通 8051：0～420MHz
- ●时钟：外部晶体或内部 RC 振荡器可选，在 ISP 下载编程用户程序时设置
- ●12K/10K/8K/6K/4K/2K 字节片内 Flash 程序存储器，擦写次数 10 万次以上
- ●512 字节片内 RAM 数据存储器
- ●芯片内 EEPROM 功能
- ●ISP / IAP，在系统可编程 / 在应用可编程,无需编程器 / 仿真器
- ●10 位 ADC，8 通道,STC12C2052AD 系列为 8 位 ADC。4 路 PWM 还可当 4 路 D/A 使用
- ●4 通道捕获 / 比较单元（PWM/PCA/CCU），STC12C2052AD 系列为 2 通道
  --- 也可用来再实现 4 个定时器或 4 个外部中断（支持上升沿 / 下降沿中断）
- ●6 个 16 位定时器，兼容普通 8051 的定时器 T0/T1,4 路 PCA 也是 4 个定时器
- ●硬件看门狗（WDT）
- ●高速 SPI 通信端口
- ●全双工异步串行口(UART),兼容普通 8051 的串口
- ●先进的指令集结构，兼容普通 8051 指令集
  4 组 8 个 8 位通用工作寄存器（共 32 个通用寄存器）
  有硬件乘法 / 除法指令
- ●通用 I/O 口（27/23/15 个），复位后为： 准双向口 / 弱上拉（普通 8051 传统 I/O 口）
  可设置成四种模式：准双向口 / 弱上拉，推挽 / 强上拉，仅为输入 / 高阻，开漏
  每个 I/O 口驱动能力均可达到 20mA，但整个芯片最大不得超过 55mA

STC12C1052，人民币 3.8 元

选择 STC12C5410AD 系列单片机的理由：

★加密性强,无法解密

★超强抗干扰:
1、高抗静电（ESD 保护）
2、轻松过 4KV 快速脉冲干扰(EFT 测试)
3、宽电压，不怕电源抖动
4、宽温度范围,-40℃～85℃

★1 个时钟 / 机器周期，可用低频晶振，大幅降低 EMI
--- 出口欧美的有力保证

★ 超低功耗:
1、掉电模式： 典型功耗 <0.1μA
2、空闲模式： 典型功耗 1.8mA
3、正常工作模式： 典型功耗 2.7mA - 7mA
4、掉电模式可由外部中断唤醒，适用于电池供电系统，如水表、气表、便携设备等。

★ 所有封装均符合欧盟 RoHS 要求,LQFP32 更可满足 Green 标准

★在系统可编程，无需编程器，无需仿真器，可远程升级

★可送 STC-ISP 下载编程器,1 万片 / 人 / 天

★内部集成 MAX810 专用复位电路，原复位电路可以保留，也可以不用，不用时 RESET 脚接 1K 电阻到地。

STC micro™
宏 晶 科 技
8051 单片机全球第一品牌
中国本土 MCU 领航者
新客户请直接联系深圳以获得更好的技术支持和服务

网址：www.MCU-Memory.com　技术支持：13922805190

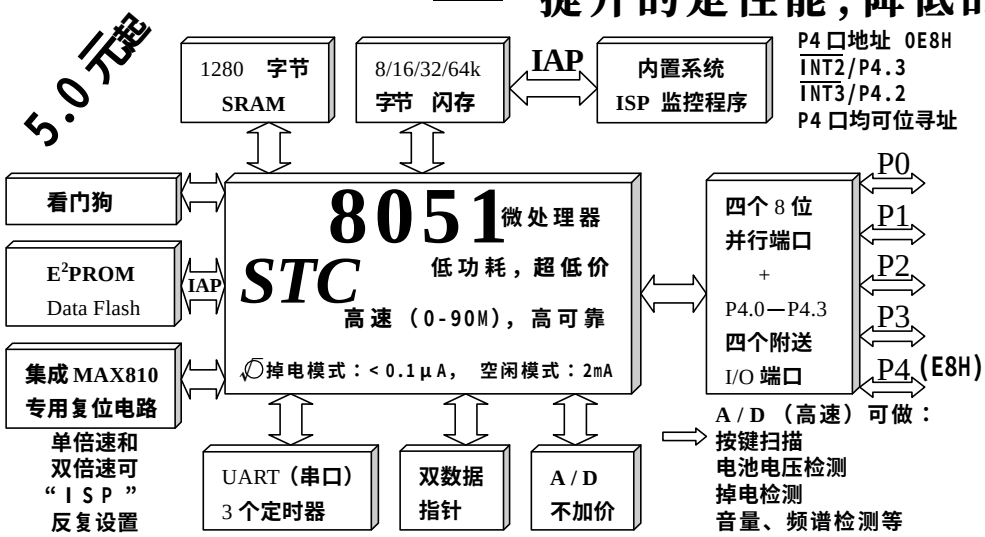
深　圳:Tel：0755-82948411　82948412　Fax：0755-82944243　82905966
广州办:Tel：020-87501705　85518657　Fax：020-85517881
上海办:Tel：021-53560136　53560138　Fax：021-53080587
北京办:Tel：010-62538687　62634001　Fax：010-62538683

免费索取

从网上下载样品申请单,
传真至深圳申请 STC 单片机
样片及 ISP 下载线 / 编程工具

STC 89系列单片机，高速、高可靠、在线编程

—— 提升的是性能，降低的是成本



选择STC89C52RC系列 STC89C58RD+系列单片机的理由：

★加密性强，无法解密

★超强抗干扰：

1、高抗静电（ESD保护）
2、轻松过2KV/4KV快速脉冲干扰(EFT测试)
3、宽电压，不怕电源抖动
4、宽温度范围，-40℃～85℃

★三大降低单片机时钟对外部电磁辐射的措施：

—— 出口欧美的有力保证

1、禁止ALE输出；
2、如选6时钟/机器周期,外部时钟频率可降一半；
3、单片机时钟振荡器增益可设为1/2gain。

★超低功耗：

1、掉电模式： 典型功耗 <0.1μA
2、正常工作模式： 典型功耗 4mA - 7mA
3、掉电模式可由外部中断唤醒，适用于电池供电系统，如水表、气表、便携设备等。

★在系统可编程，无需编程器，无需仿真器

★可送STC-ISP下载编程器，1万片/人/天

★可供应内部集成MAX810专用复位电路的单片机，只有D版本才有内部集成专用复位电路，原复位电路可以保留，也可以不用，不用时RESET脚接1K电阻到地

STC 89系列单片机选型一览表　超低价

型号	最高时钟频率Hz 5V	最高时钟频率Hz 3V	Flash存储器	RAM字节	降低EMI	看门狗	双倍速	P4口	ISP	IAP	E²PROM字节	A/D
STC 89C51 RC	0～80M		4K	512	√	✆	√	√	√	√	2K	
STC 89C52 RC	0～80M		8K	512	√	✆	√	√	√	√	2K	
STC 89C53 RC	0～80M		15K	512	√	✆	√	√	√	√		
STC 89C54 RD+	0～80M		16K	1280	√	✆	√	√	√	√	16K	
STC 89C55 RD+	0～80M		20K	1280	√	✆	√	√	√	√	16K	
STC 89C58 RD+	0～80M		32K	1280	√	✆	√	√	√	√	16K	
STC 89C516 RD+	0～80M		64K	1280	√	✆	√	√	√	√		
STC 89LE51 RC		0～80M	4K	512	√	✆	√	√	√	√	2K	
STC 89LE52 RC		0～80M	8K	512	√	✆	√	√	√	√	2K	
STC 89LE53 RC		0～80M	15K	512	√	✆	√	√	√	√		
STC 89LE54 RD+		0～80M	16K	1280	√	✆	√	√	√	√	16K	
STC 89LE58 RD+		0～80M	32K	1280	√	✆	√	√	√	√	16K	
STC 89LE516RD+		0～80M	64K	1280	√	✆	√	√	√	√		
STC 89LE516AD	0～90M，3.6～2.4V		64K	512	√		√	√	√			√

关于单片机说明：<管脚与流行的8051兼容>　大客户超低价

DIP-40，PLCC-44，LQFP-44封装(RC/RD+系列PLCC、LQFP有P4口地址E8H,AD系列P4口为C0H)

RC/RD+系列PLCC、LQFP多两个外部中断P4.2/INT3，P4.3/INT2。 P4口均可位寻址

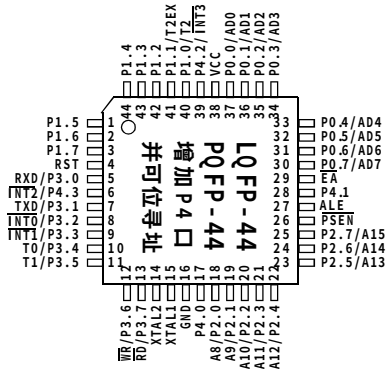
5V：5.5V～3.8V；3V：3.8V～2.4V(仅针对RC/RD+系列)

✆ 真正的看门狗，可放心省去外部看门狗，缺省为关闭，打开后无法关闭。单倍速和双倍速可反复设置

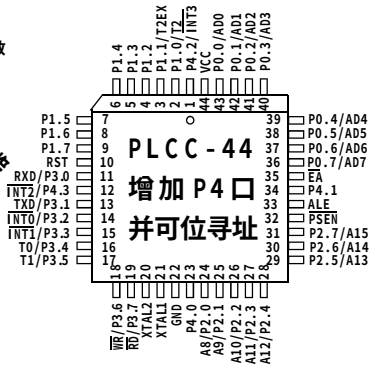
✆ “6时钟/机器周期”和“12时钟/机器周期”可在ISP编程时反复设置，新的设置冷启动后才生效

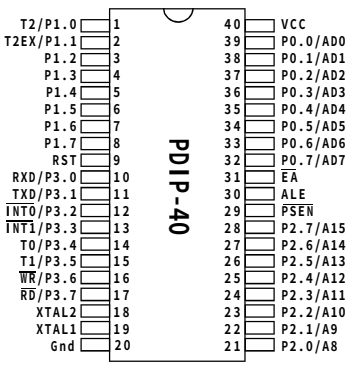
另 STC89LE516AD、58AD、54AD、52AD、51AD系列单片机，带高速A/D转换

推荐流行的LQFP44 超小贴片封装

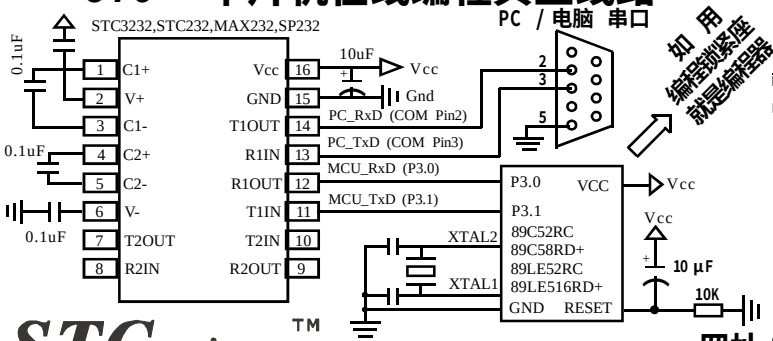


传统的PLCC和PDIP封装稳定供货





STC 单片机在线编程典型线路



另有：STC89LE516AD系列单片机

网址：www.MCU-Memory.com　技术支持：13922805190

STC micro™
宏晶科技
8051单片机全球第一品牌
中国本土MCU领航者
新客户请直接联系深圳以获得更好的技术支持和服务

深　圳：Tel：0755-82948411	82948412	Fax：0755-82944243	82905966	
广州办：Tel：020-87501705	85518657	Fax：020-85517881		
上海办：Tel：021-53560136	53560138	Fax：021-53080587		
北京办：Tel：010-62538687	62634001	Fax：010-62538683		

免费索取
从网上下载样品申请单，传真至深圳申请STC单片机样片及ISP下载线/编程工具

目录

第二章 STC12 系列单片机总体介绍

2.1 STC12C5410AD / 2052AD 系列 1T 单片机简介

STC12C5410AD 系列及 STC12C2052AD 系列单片机是宏晶科技生产的单时钟 / 机器周期(1T)的单片机，是高速 / 低功耗 / 超强抗干扰的新一代 8051 单片机，指令代码完全兼容传统 8051,但速度快 8-12 倍，内部集成 MAX810 专用复位电路。4 路 PWM,8 路高速 10 位 A/D 转换,针对电机控制，强干扰场合。

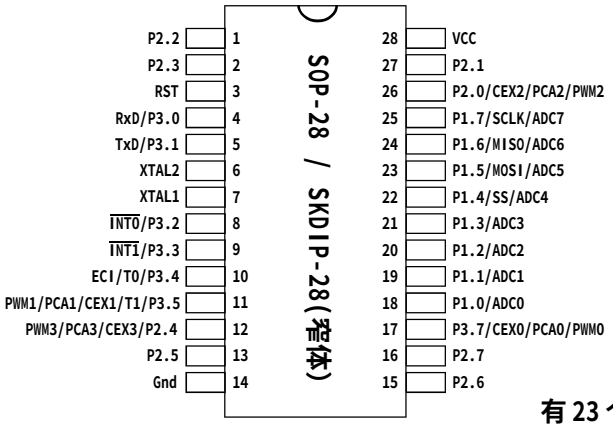
STC12C2052AD 系列只有 2 路 PWM,8 路高速 8 位 A/D 转换。

1. 增强型 8051 CPU，1T，单时钟 / 机器周期，指令代码完全兼容传统 8051
2. 工作电压：
   STC12C5410AD 系列工作电压：5.5V - 3.5V（5V 单片机）/ 3.8V - 2.2V（3V 单片机）
   STC12C2052AD 系列工作电压：5.5V - 3.5V（5V 单片机）/ 3.8V - 2.2V（3V 单片机）
3. 工作频率范围：0 - 35 MHz，相当于普通 8051 的 0～420MHz
4. 用户应用程序空间 1K / 2K / 4K / 6K / 8K / 10K / 12K 字节.....
5. 片上集成 512 字节 RAM(STC12C5410AD 系列)，STC12C2052AD 系列单片机为 256 字节 RAM
6. 通用 I/O 口（27/23/15 个），复位后为： 准双向口 / 弱上拉（普通 8051 传统 I/O 口）
   可设置成四种模式：准双向口 / 弱上拉，推挽 / 强上拉，仅为输入 / 高阻，开漏
   每个 I/O 口驱动能力均可达到 20mA，但整个芯片最大不得超过 55mA
7. ISP（在系统可编程）/ IAP（在应用可编程），无需专用编程器，无需专用仿真器
   可通过串口（P3.0/P3.1）直接下载用户程序，数秒即可完成一片
8. EEPROM 功能
9. 看门狗
10. 内部集成 MAX810 专用复位电路（外部晶体 12M 以下时，可省外部复位电路）
11. 时钟源：外部高精度晶体 / 时钟，内部 R/C 振荡器
    用户在下载用户程序时，可选择是使用内部 R/C 振荡器还是外部晶体 / 时钟
    常温下内部 R/C 振荡器频率为：5.2MHz ～ 6.8MHz
    精度要求不高时，可选择使用内部时钟，但因为有制造误差和温漂，应认为是 4MHz ～ 8MHz
12. 共 6 个 16 位定时器 / 计数器，
    两个专用 16 位定时器 T0 和 T1
    再加上 PCA 模块可再实现 4 个 16 位定时器，STC12C2052AD 系列只有两路 PCA
13. 外部中断 2 路,下降沿中断或低电平触发中断,Power Down 模式可由外部中断唤醒
14. PWM(4 路) /PCA（可编程计数器阵列,4 路),5410 系列是 4 路，2052 系列只有两路
    --- 也可用来当 4 路 D/A 使用
    --- 也可用来再实现 4 个定时器
    --- 也可用来再实现 4 个外部中断(上升沿中断 / 下降沿中断均可分别或同时支持)
15. A/D 转换，10 位精度 ADC，共 8 路。STC12C2052AD 系列只有 8 位精度
16. 通用全双工异步串行口(UART)，由于 STC12 系列是高速的 8051，也可再用定时器软件实现多串口
17. SPI 同步通信口，主模式 / 从模式
18. 工作温度范围： 0 - 75℃ / -40 - +85℃
19. 封装：LQFP-32,SOP-32/28/20， SKDIP-28，PDIP-20，TSSOP-20(超小封状 6.4mm × 6.4mm，定货)
    LQFP/SOP32 有 27 个 I/O 口，SOP28/SKDIP28 有 23 个 I/O 口，SOP20/TSSOP20/PDIP20 有 15 个 I/O 口，
    I/O 口不够时，可用 2 到 3 根普通 I/O 口线外接 74HC164/165/595（均可级联）来扩展 I/O 口，
    还可用 A/D 做按键扫描来节省 I/O 口，或用双 CPU,三线通信，还多了串口。

2.2 STC12 系列单片机管脚图及封装尺寸图

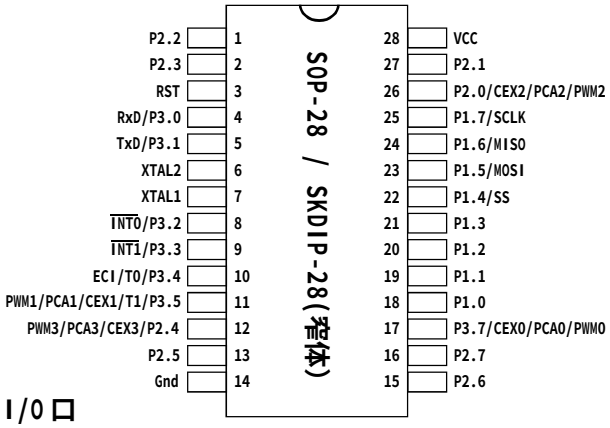
2.2.1 管脚图(所有封装形式均满足欧盟 RoHS 要求，LQFP-32 采用 Green 标准生产)

强烈推荐选择 SOP-20/28/32 贴片封装,尽量不选落后的插件 DIP 封装

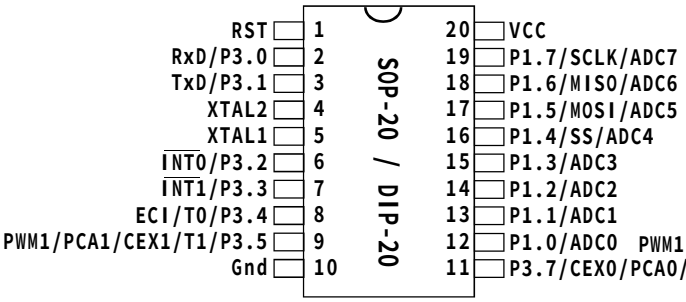


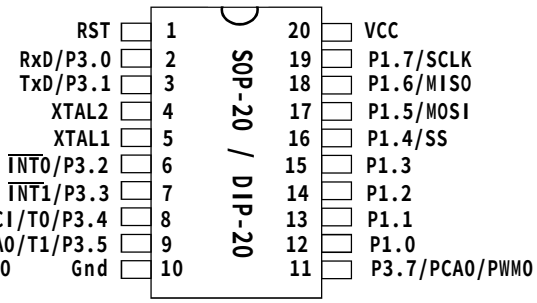
有 23 个 I/O 口

STC12C5410AD 系列（有 A/D 转换），28-Pin



STC12C5410 系列（无 A/D 转换），28-Pin





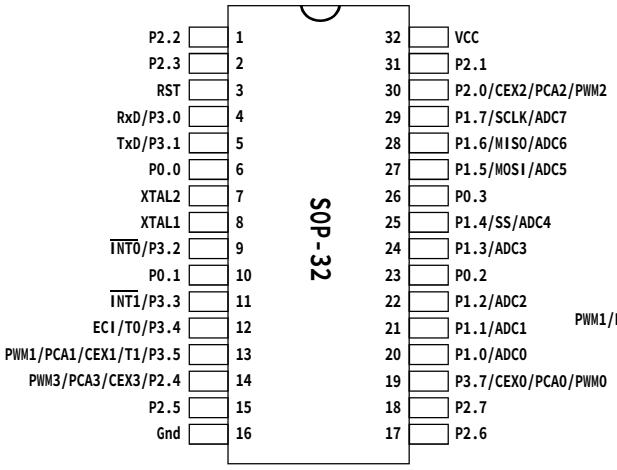
有 15 个 I/O 口

STC12C5410AD 系列（有 A/D 转换），20-Pin

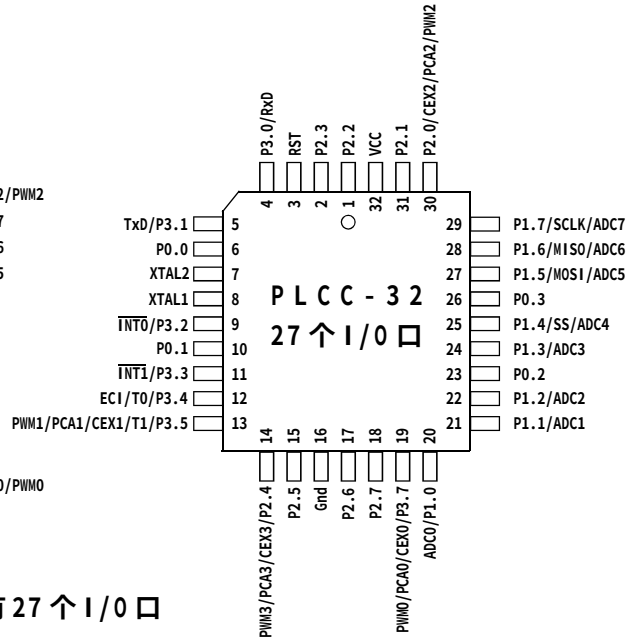
STC12C2052AD 系列（有 A/D 转换），20-Pin

STC12C5410 系列（无 A/D 转换），20-Pin

STC12C2052 系列（无 A/D 转换），20-Pin



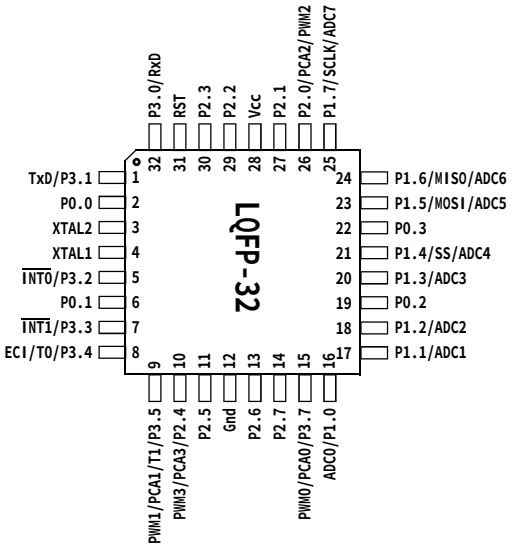
STC12C5410AD 系列（有 A/D 转换），32-Pin，有 27 个 I/O 口



不推荐选 PLCC-32

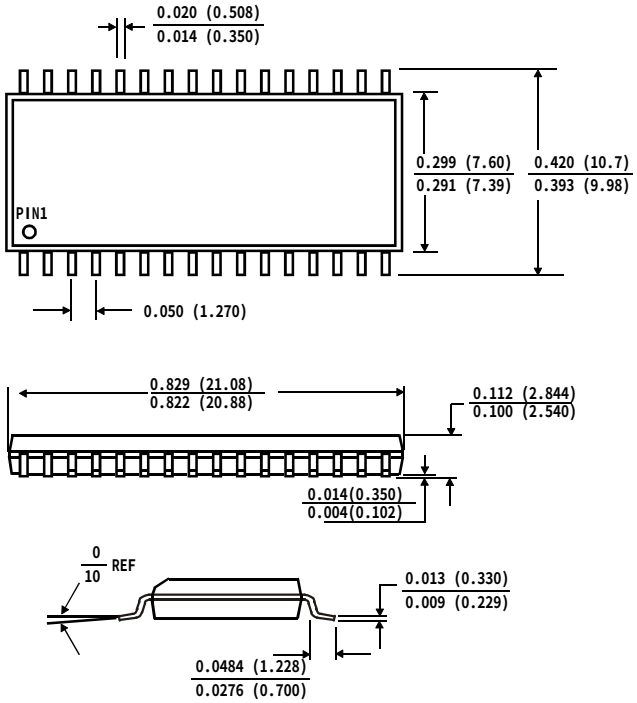
如须特别满足更高层次的 Green 标准，请采用 LQFP-32 封装

长 x 宽 = 9mm x 9mm，高 < 1.6mm

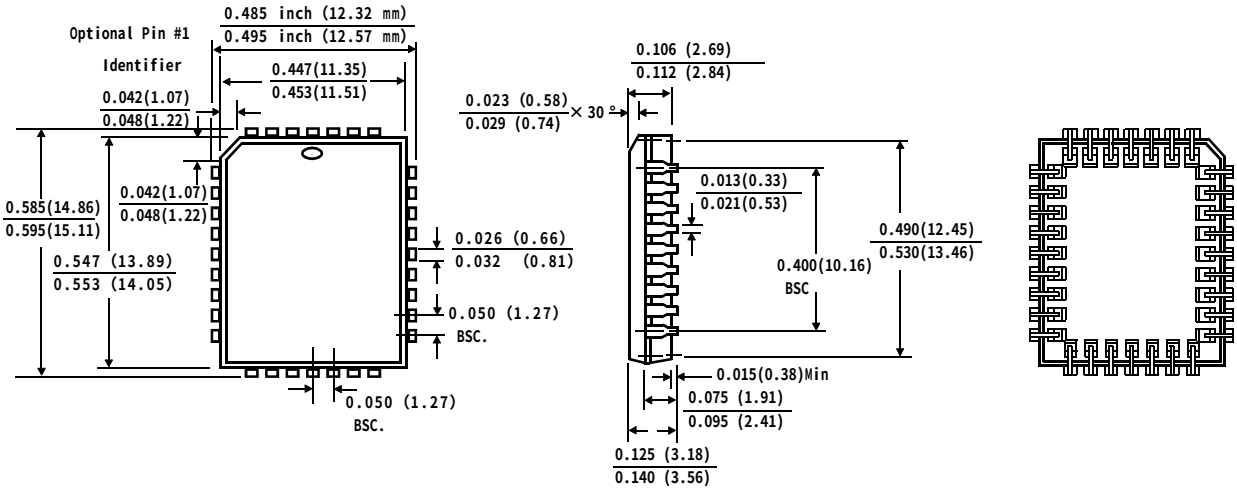


2.2.2 封装尺寸图

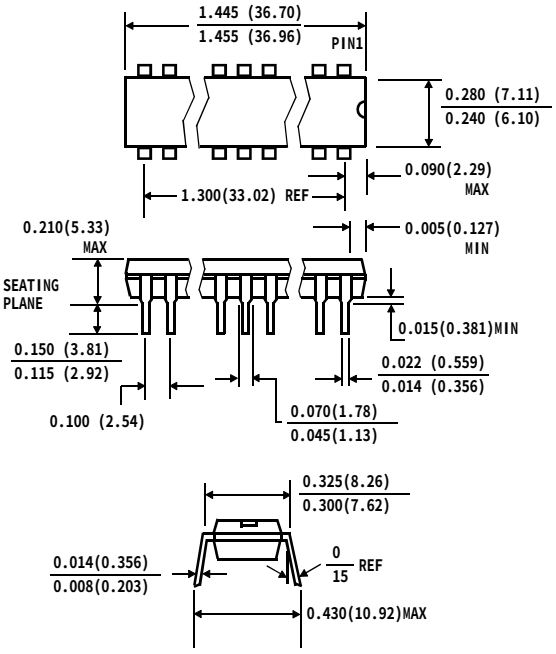
32-PIN SMALL OUTLINE PACKAGE (SOP-32)



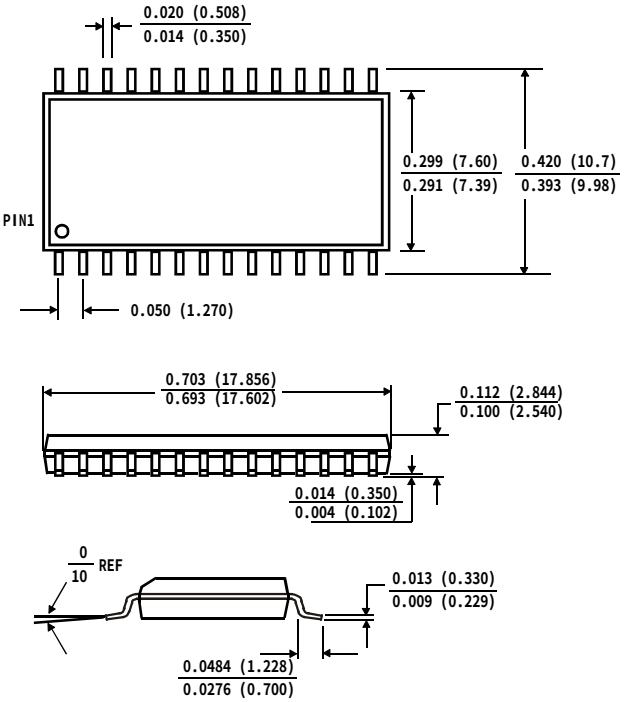
32-PIN PLASTIC LEAD CHIP CARRIER (PLCC-32)



28-PIN PLASTIC DUAL-IN-LINE PACKAGE (SKDIP-28)



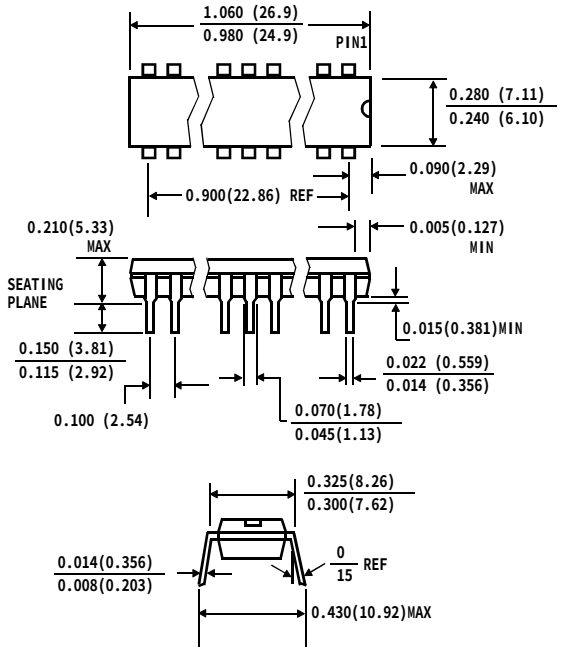
28-PIN SMALL OUTLINE PACKAGE (SOP-28)



20P3, 20-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP-20)

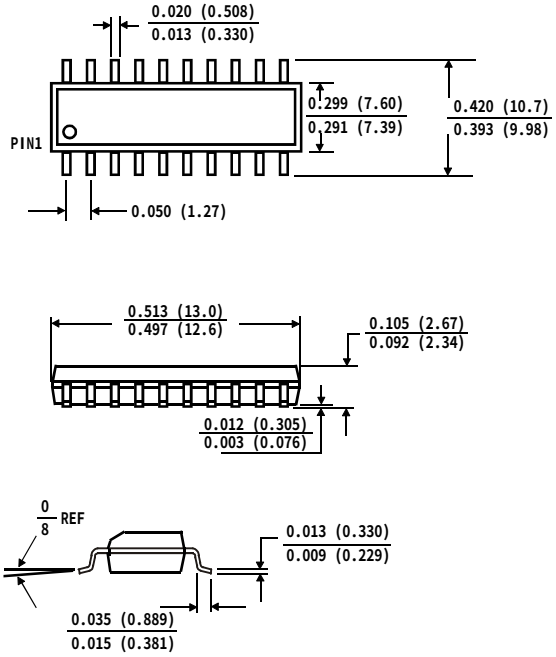
Dimensions in Inches and (Millimeters)

JEDEC STANDARD MS-001 AD



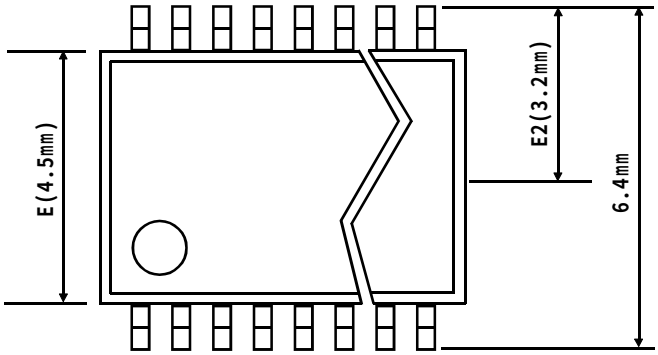
20S, 20-lead, 0.300" Wide, Plastic Gull WIng Small Outline (SOIC-20 / SOP-20)

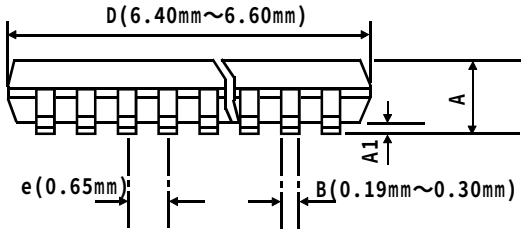
Dimensions in Inches and (Millimeters)

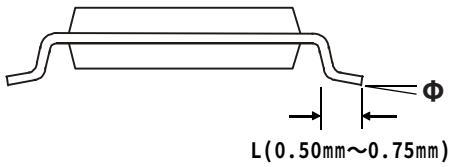


STC12C5410AD / 2052AD 系列的超小封装TSSOP-20(仅为6.4mm x 6.4mm)
----20-Pin TSSOP-20封装，尺寸只有常规的SOP-8大小

PACKAGE：PLASTIC SHRINK SMALL OUTLINE（TSSOP-20，6.4mm×6.4mm）

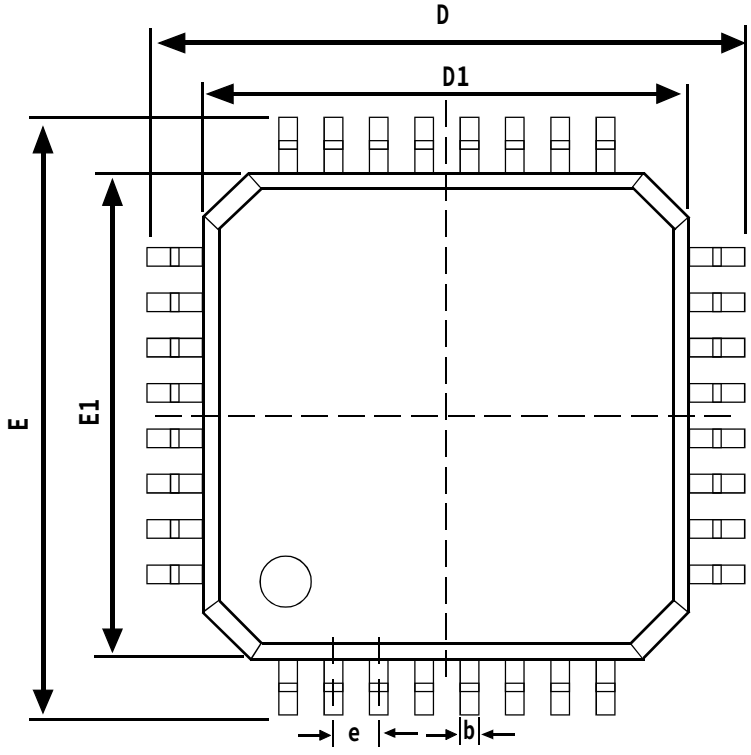


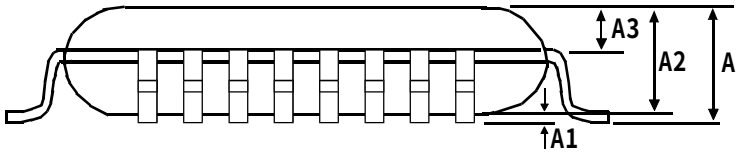


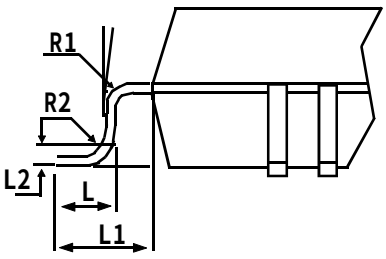


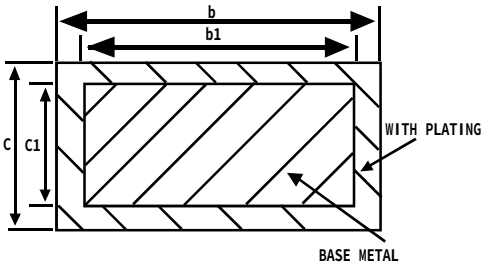
DIMENSIONS in inches (mm) Minimum/Maximum	20-PIN
A	- /0.043 (- /1.10)
A1	0.002/0.006 (0.05/0.15mm)
B	0.007/0.012 (0.19/0.30mm)
D	0.252/0.260 (6.40/6.60mm)
E	0.169/0.177 (4.30/4.50mm)
e	0.026 BSC (0.65mm BSC)
E2	0.126 BSC (3.20mm BSC)
L	0.020/0.030 (0.50/0.75mm)
Φ	0°/8°

LQFP-32 OUTLINE PACKAGE









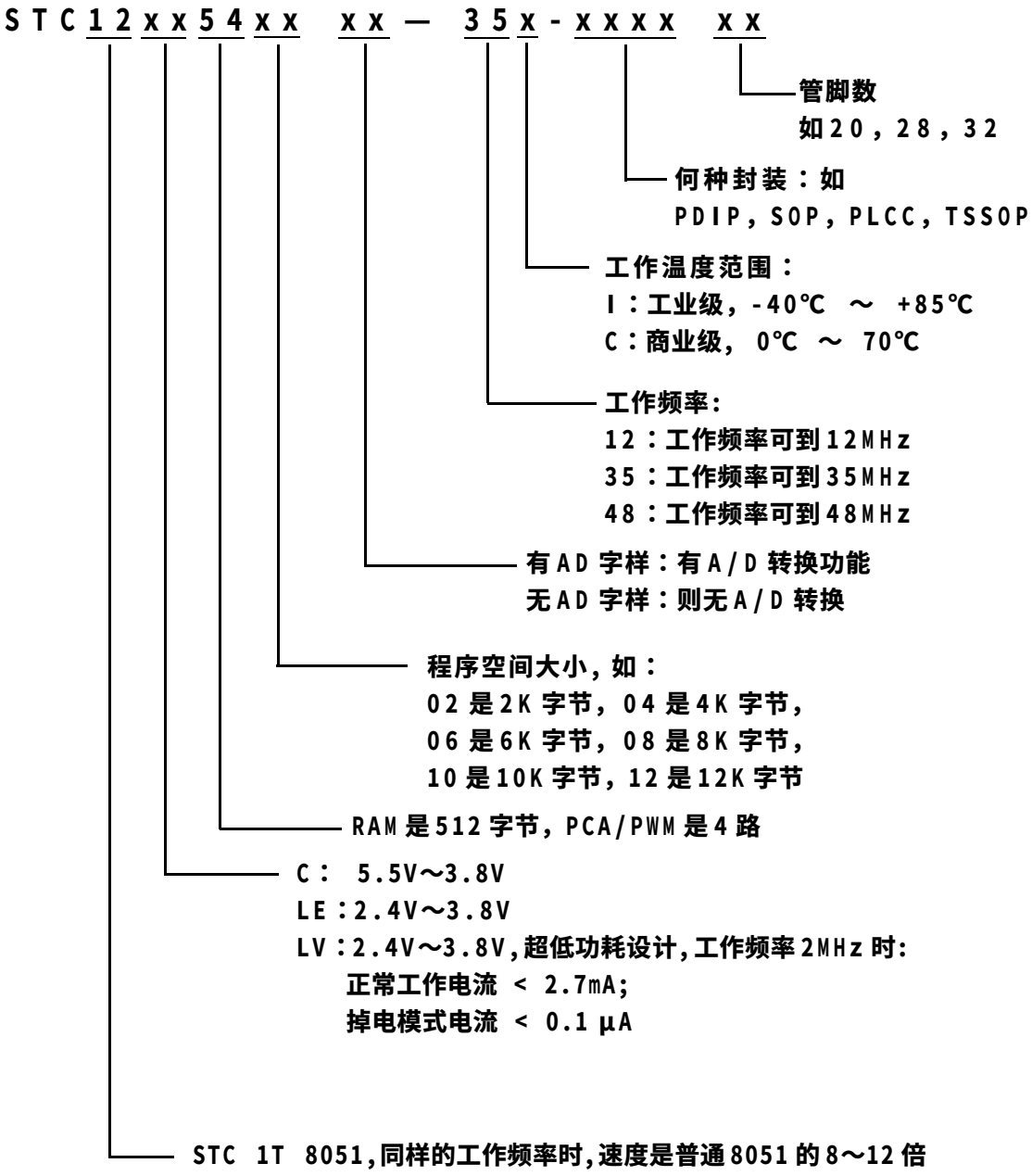
COMMON DIMENSIONS			
(UNITS OF MEASURE = MILLIMETER)			
SYMBOL	MIN	NOM	MAX
A	-	-	1.60
A1	0.05	-	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.32	-	0.43
b1	0.31	0.35	0.39
c	0.13	-	0.18
c1	0.12	0.127	0.134
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
e	0.80BSC		
L	0.45	0.60	0.75
L1	1.00REF		
L2	0.25BSC		
R1	0.08	-	-
R2	0.08	-	0.20
S	0.20	-	-
Φ	0°	3.5°	7°
Φ1	0°	–	–
Φ2	11°	12°	13°
Φ3	11°	12°	13°

2.3 STC12C5410AD / 2052AD 系列单片机选型一览表

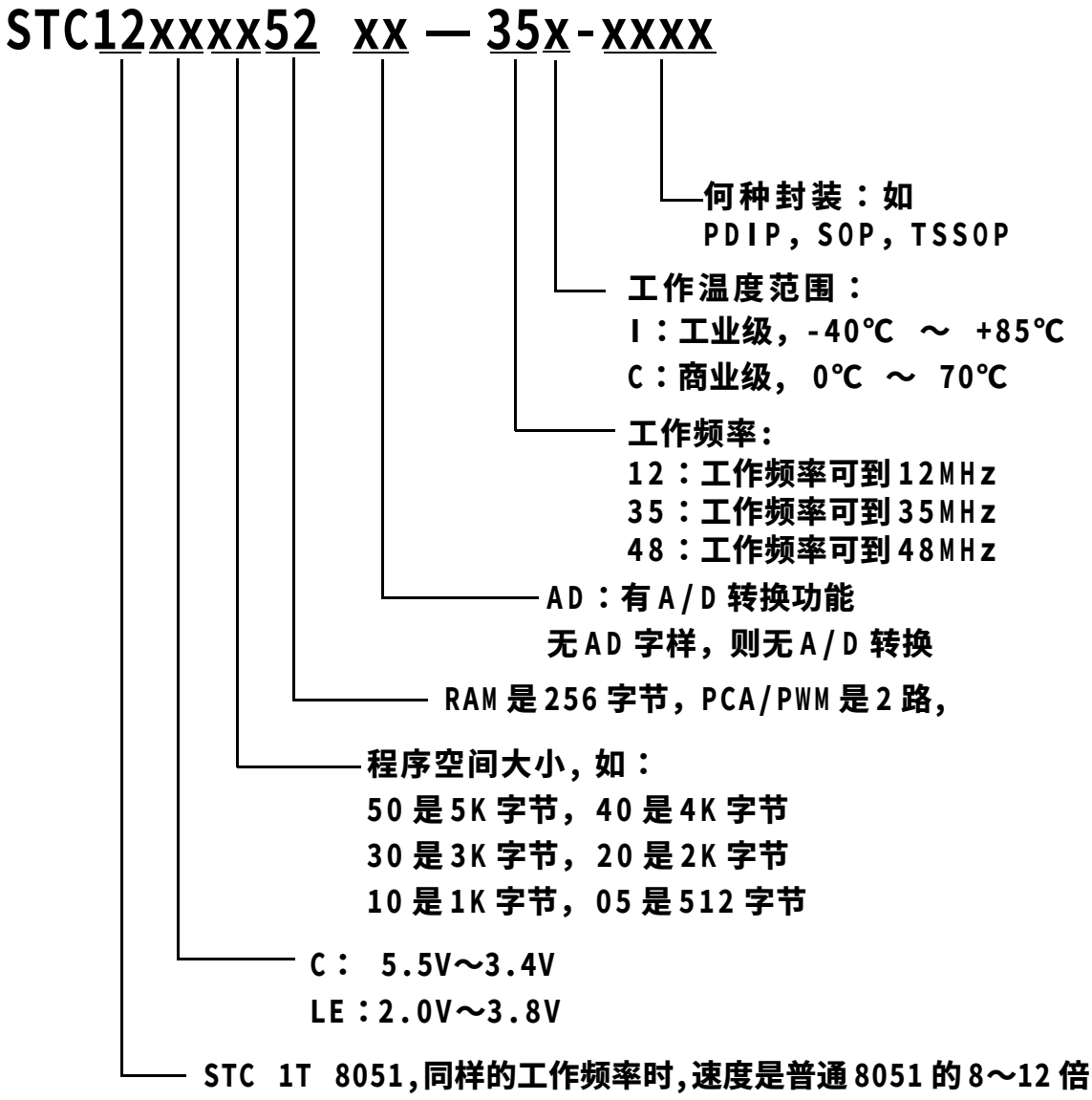
型 号	工作电压(V)	Flash程序存储器字节	SRAM字节	定时器	UART	PCA 16位PWM 8位	A/D 8路	I/O	看门狗	内置复位	EEPROM	SPI	封装 20-Pin	封装 28-Pin	封装 32-Pin
STC12C2052AD系列单片机选型一览															
STC12C1052	5.5 - 3.5	1K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP	管脚兼容 89C2051 超强抗干扰 无法解密	
STC12C1052AD	5.5 - 3.5	1K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12C2052	5.5 - 3.5	2K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12C2052AD	5.5 - 3.5	2K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12C4052	5.5 - 3.5	4K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12C4052AD	5.5 - 3.5	4K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12C5052	5.5 - 3.5	5K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12C5052AD	5.5 - 3.5	5K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE1052	2.2 - 3.8	1K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE1052AD	2.2 - 3.8	1K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE2052	2.2 - 3.8	2K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE2052AD	2.2 - 3.8	2K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE4052	2.2 - 3.8	4K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE4052AD	2.2 - 3.8	4K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE5052	2.2 - 3.8	5K	256	4	有	2路		15	有	有	有	有	SOP/TSSOP/DIP		
STC12LE5052AD	2.2 - 3.8	5K	256	4	有	2路	8位	15	有	有	有	有	SOP/TSSOP/DIP		
STC12C5410AD系列单片机选型一览															
STC12C5402	5.5 - 3.5	2K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5402AD	5.5 - 3.5	2K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5404	5.5 - 3.5	4K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5404AD	5.5 - 3.5	4K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5406	5.5 - 3.5	6K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5406AD	5.5 - 3.5	6K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5408	5.5 - 3.5	8K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5408AD	5.5 - 3.5	8K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5410	5.5 - 3.5	10K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5410AD	5.5 - 3.5	10K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5412	5.5 - 3.5	12K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12C5412AD	5.5 - 3.5	12K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
⋮															
STC12LE5402	2.2 - 3.8	2K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5402AD	2.2 - 3.8	2K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5404	2.2 - 3.8	4K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5404AD	2.2 - 3.8	4K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5406	2.2 - 3.8	6K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5406AD	2.2 - 3.8	6K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5408	2.2 - 3.8	8K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5408AD	2.2 - 3.8	8K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5410	2.2 - 3.8	10K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5410AD	2.2 - 3.8	10K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5412	2.2 - 3.8	12K	512	6	有	4路		27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
STC12LE5412AD	2.2 - 3.8	12K	512	6	有	4路	10位	27/23	有	有	有	有	SOP/TSSOP/DIP	SOP/SKDIP	SOP/LQFP
⋮															

2.4 STC12C5410AD 及 STC12C2052AD 系列单片机命名规则

STC12C5410AD 系列单片机命名规则



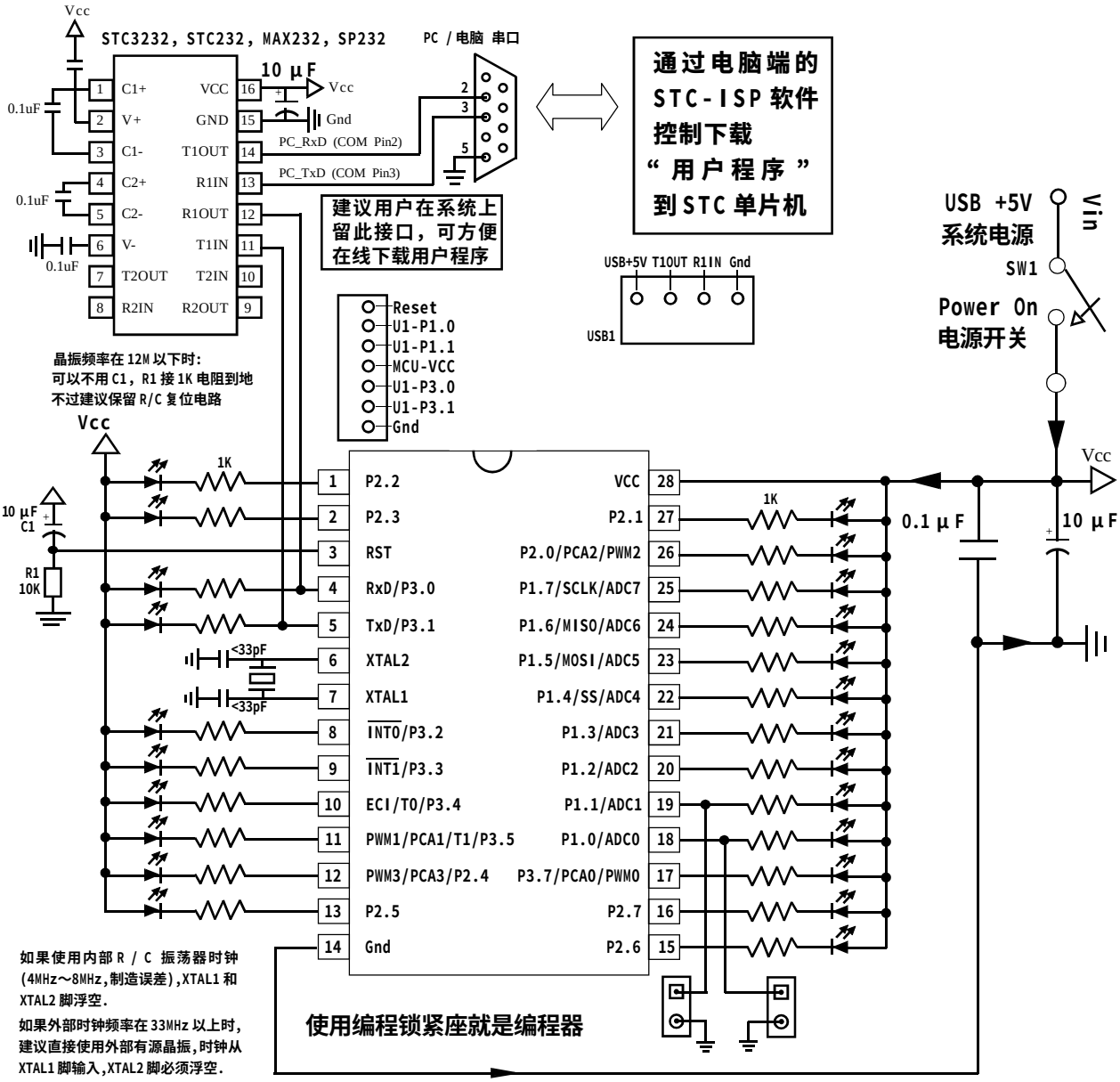
STC12C2052AD 系列单片机命名规则



2.5 STC12C5410AD / 2052AD 系列单片机典型应用电路

---- 通过 RS-232 转换器连接电脑就可以下载程序

2.5.1 STC12C5410AD 系列单片机 28 脚典型应用电路



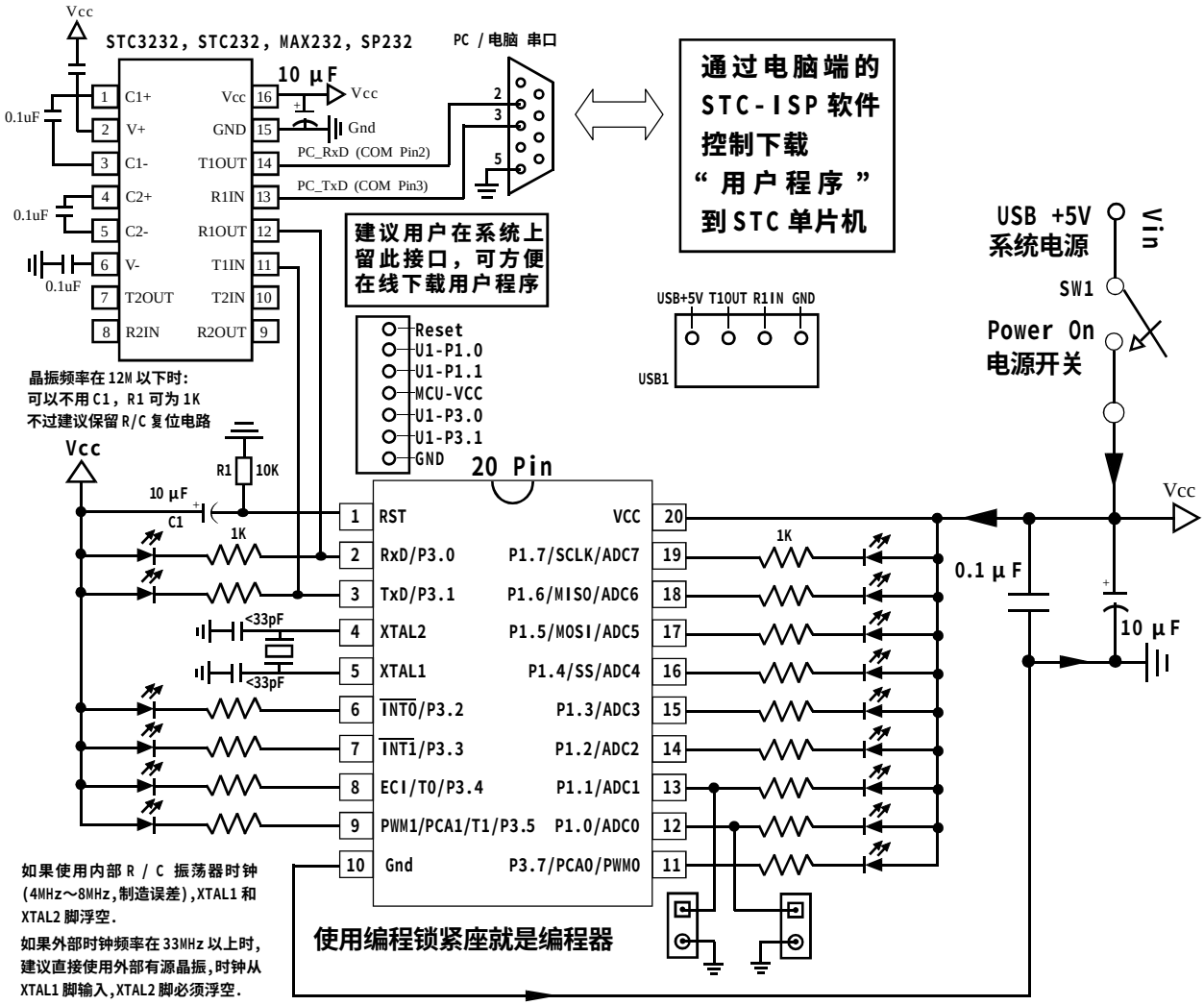
此线路已做成一个 STC12C5410AD 系列单片机 ISP 下载编程工具，可直接赠送给客户

用户在自己的目标系统上，如将 P3.0/P3.1 经过 RS-232 电平转换器转换后连接到电脑的普通 RS-232 串口，就可以在系统编程 / 升级用户软件。 建议如果用户板上无 RS-232 电平转换器， 应引出一个插座，含 Gnd / P3.1 / P3.0 / Vcc 四个信号线，这样就可以在用户系统上直接编程了。当然如能引出 Gnd / P3.1 / P3.0 / Vcc / P1.1 / P1.0 六个信号线为好，因为可以通过 P1.0/P1.1 禁止 ISP 下载程序。如果能将 Gnd / P3.1 / P3.0 / Vcc / P1.1 / P1.0 / Reset 七个信号线引出就更好了，这样可以很方便的使用“脱机下载板（无需电脑）”。

关于 ISP 编程的原理及应用指南详见“STC12C5410AD 系列单片机开发 / 编程工具说明”部分。另外我们有标准化的编程下载工具，用户可以在上面编程后再插到目标系统上，也可以借用它上面的 RS-232 电平转换器连接到电脑，以做下载编程之用。编程一个芯片大致需几秒钟，速度比普通的通用编程器快很多，故无须买第三方的高价编程器。

电脑端 STC-ISP 软件从网站 www.MCU-Memory.com 下载

2.5.2 STC12C5410AD 系列及 STC12C2052AD 系列单片机 20 脚典型应用电路



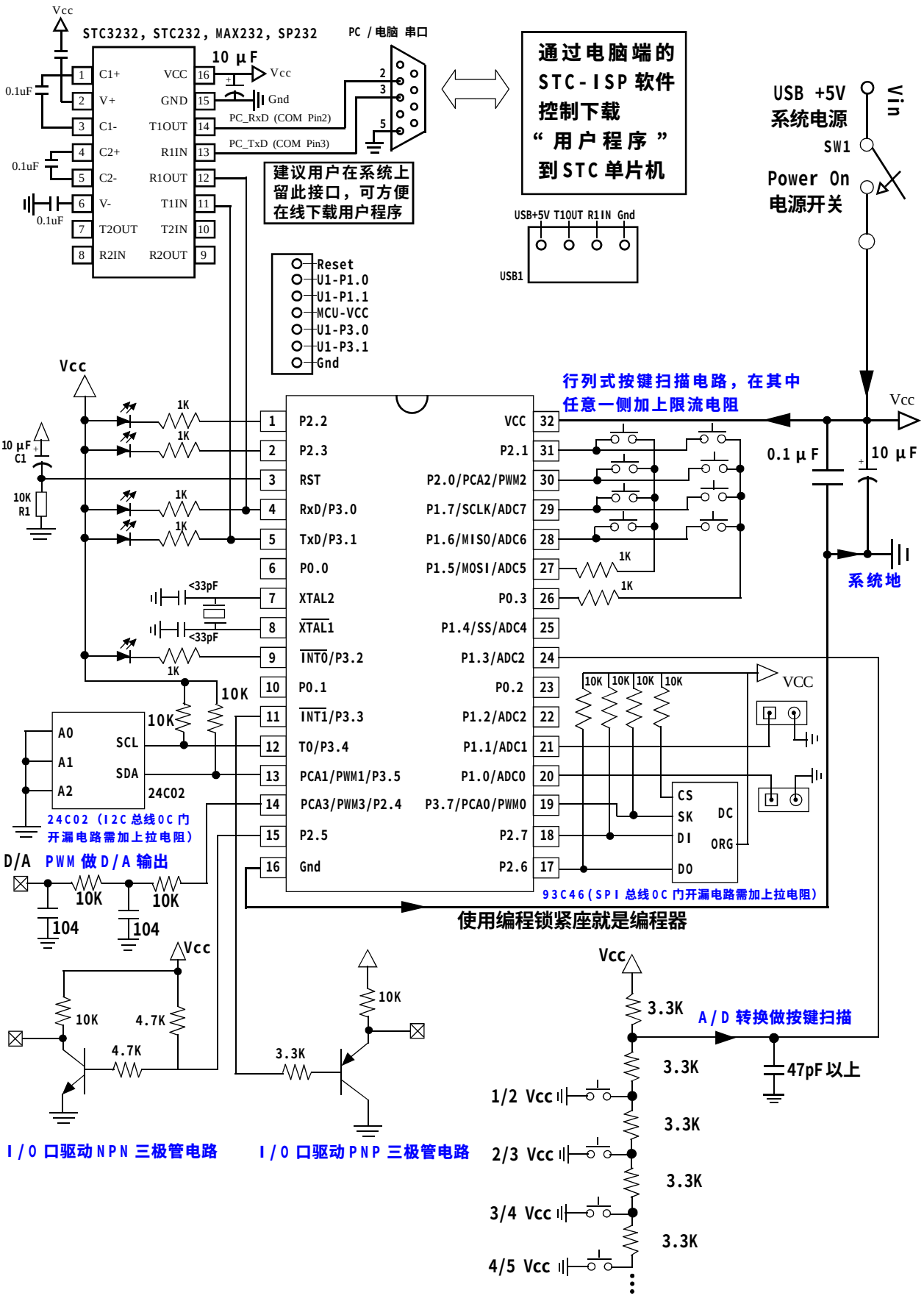
此线路已做成一个 STC12C5410AD 系列单片机 ISP 下载编程工具，可直接赠送给客户

用户在自己的目标系统上，如将 P3.0/P3.1 经过 RS-232 电平转换器转换后连接到电脑的普通 RS-232 串口，就可以在系统编程 / 升级用户软件。建议如果用户板上无 RS-232 电平转换器，应引出一个插座，含 Gnd / P3.1 / P3.0 / Vcc 四个信号线，这样就可以在用户系统上直接编程了。当然如能引出 Gnd / P3.1 / P3.0 / Vcc / P1.1 / P1.0 六个信号线为好，因为可以通过 P1.0/P1.1 禁止 ISP 下载程序。如果能将 Gnd / P3.1 / P3.0 / Vcc / P1.1 / P1.0 / Reset 七个信号线引出就更好了，这样可以很方便的使用“脱机下载板（无需电脑）”。

关于 ISP 编程的原理及应用指南详见“STC12C5410AD 系列单片机开发 / 编程工具说明”部分。另外我们有标准化的编程下载工具，用户可以在上面编程后再插到目标系统上，也可以借用它上面的 RS-232 电平转换器连接到电脑，以做下载编程之用。编程一个芯片大致需几秒钟，速度比普通的通用编程器快很多，故无须买第三方的高价编程器。

电脑端 STC-ISP 软件从网站 www.MCU-Memory.com 下载

2.5.3 STC12C5410AD 系列单片机 32 脚综合应用电路



2.6 指令系统分类总结及与普通 8051 指令执行时间对比

--- 与 8051 指令代码完全兼容，但执行的时间效率大幅提升

--- 其中 INC DPTR 指令的执行速度大幅提升 24 倍

--- 共有 12 条指令，一个时钟就可以执行完成，平均速度快 8～12 倍

如果按功能分类，STC89/12 系列单片机指令系统可分为：

1.数据传送类指令；

2.算术操作类指令；

3.逻辑操作类指令；

4.控制转移类指令；

5.布尔变量操作类指令。

按功能分类的指令系统表如下表所示。

传统 12T 的 8051 指令执行所需时钟

STC12C5410AD 系列指令执行所需时钟

数据传送类指令

助记符		功能说明	字节数	12时钟/机器周期所需时钟	1时钟/机器周期所需时钟	效率提升
MOV	A，Rn	寄存器内容送入累加器	1	12	1	12倍
MOV	A，direct	直接地址单元中的数据送入累加器	2	12	2	6倍
MOV	A，@Ri	间接RAM中的数据送入累加器	1	12	2	6倍
MOV	A，#data	立即送入累加器	2	12	2	6倍
MOV	Rn，A	累加器内容送入寄存器	1	12	2	6倍
MOV	Rn，direct	直接地址单元中的数据送入寄存器	2	24	4	6倍
MOV	Rn，#data	立即数送入寄存器	2	12	2	6倍
MOV	direct，A	累加器内容送入直接地址单元	2	12	3	4倍
MOV	direct，Rn	寄存器内容送入直接地址单元	2	24	3	8倍
MOV	direct，direct	直接地址单元中的数据送入另一个直接地址单元	3	24	4	6倍
MOV	direct，@Ri	间接RAM中的数据送入直接地址单元	2	24	4	6倍
MOV	direct，#data	立即数送入直接地址单元	3	24	3	8倍
MOV	@Ri，A	累加器内容送间接RAM单元	1	12	3	4倍
MOV	@Ri，direct	直接地址单元数据送入间接RAM单元	2	24	3	8倍
MOV	@Ri，#data	立即数送入间接RAM单元	2	12	3	4倍
MOV	DPTR，#data16	16位立即数送入地址寄存器	3	24	3	8倍
MOVC	A,@A+DPTR	以DPTR为基地址变址寻址单元中的数据送入累加器	1	24	4	6倍
MOVC	A,@A+PC	以PC为基地址变址寻址单元中的数据送入累加器	1	24	4	6倍
MOVX	A,@Ri	外部RAM（8位地址）送入累加器	1	24	4	6倍
MOVX	A,@DPTR	外部RAM（16位地址）送入累加器	1	24	3	8倍
MOVX	@Ri,A	累加器送外部RAM（8位地址）	1	24	3	8倍
MOVX	@DPTR ,A	累加器送外部RAM（16位地址）	1	24	3	8倍
PUSH	direct	直接地址单元中的数据压入堆栈	2	24	4	6倍
POP	direct	出栈送直接地址单元	2	24	3	8倍
XCH	A,Rn	寄存器与累加器交换	1	12	3	4倍
XCH	A,direct	直接地址单元与累加器交换	2	12	4	3倍
XCH	A,@Ri	间接RAM与累加器交换	1	12	4	3倍
XCHD	A,@Ri	间接RAM的低半字节与累加器交换	1	12	4	3倍

算术操作类指令

助记符		功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟	提升效率
ADD	A，Rn	寄存器内容加到累加器	1	12	2	6倍
ADD	A，direct	直接地址单元中的数据加到累加器	2	12	3	4倍
ADD	A，@Ri	间接RAM中的数据加到累加器	1	12	3	4倍
ADD	A，#data	立即加到累加器	2	12	2	6倍
ADDC	A，Rn	寄存器内容带进位加到累加器	1	12	2	6倍
ADDC	A，direct	直接地址单元的内容带进位加到累加器	2	12	3	4倍
ADDC	A，@Ri	间接RAM内容带进位加到累加器	1	12	3	4倍
ADDC	A，#data	立即数带进位加到累加器	2	12	2	6倍
SUBB	A，Rn	累加器带借位减寄存器内容	1	12	2	6倍
SUBB	A，direct	累加器带借位减直接地址单元的内容	2	12	3	4倍
SUBB	A，@Ri	累加器带借位减间接RAM中的内容	1	12	3	4倍
SUBB	A，#data	累加器带借位减立即数	2	12	2	6倍
INC	A	累加器加1	1	12	2	6倍
INC	Rn	寄存器加1	1	12	3	4倍
INC	direct	直接地址单元加1	2	12	4	3倍
INC	@Ri	间接RAM单元加1	1	12	4	3倍
DEC	A	累加器减1	1	12	2	6倍
DEC	Rn	寄存器减1	1	12	3	4倍
DEC	direct	直接地址单元减1	2	12	4	3倍
DEC	@Ri	间接RAM单元减1	1	12	4	3倍
INC	DPTR	地址寄存器DPTR加1	1	24	1	24倍
MUL	AB	A乘以B	1	48	4	12倍
DIV	AB	A除以B	1	48	5	9.6倍
DA	A	累加器十进制调整	1	12	4	3倍

逻辑操作类指令

助记符		功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟	提升效率
ANL	A，Rn	累加器与寄存器相“与”	1	12	2	6倍
ANL	A，direct	累加器与直接地址单元相“与”	2	12	3	4倍
ANL	A，@Ri	累加器与间接RAM单元相“与”	1	12	3	4倍
ANL	A，#data	累加器与立即数相“与”	2	12	2	6倍
ANL	direct，A	直接地址单元与累加器相“与”	2	12	4	3倍
ANL	direct，#data	直接地址单元与立即数相“与”	3	24	4	6倍
ORL	A，Rn	累加器与寄存器相“或”	1	12	2	6倍
ORL	A，direct	累加器与直接地址单元相“或”	2	12	3	4倍
ORL	A，@Ri	累加器与间接RAM单元相“或”	1	12	3	4倍
ORL	A，#data	累加器与立即数相“或”	2	12	2	6倍
ORL	direct，A	直接地址单元与累加器相“或”	2	12	4	3倍
ORL	direct，#data	直接地址单元与立即数相“或”	3	24	4	6倍
XRL	A，Rn	累加器与寄存器相“异或”	1	12	2	6倍
XRL	A，direct	累加器与直接地址单元相“异或”	2	12	3	4倍
XRL	A，@Ri	累加器与间接RAM单元相“异或”	1	12	3	4倍
XRL	A，#data	累加器与立即数相“异或”	2	12	2	6倍
XRL	direct，A	直接地址单元与累加器相“异或”	2	12	4	3倍
XRL	direct，#data	直接地址单元与立即数相“异或”	3	24	4	6倍
CLR	A	累加器清“0”	1	12	1	12倍
CPL	A	累加器求反	1	12	2	6倍
RL	A	累加器循环左移	1	12	1	12倍
RLC	A	累加器带进位位循环左移	1	12	1	12倍
RR	A	累加器循环右移	1	12	1	12倍
RRC	A	累加器带进位位循环右移	1	12	1	12倍
SWAP	A	累加器半字节交换	1	12	1	12倍

控制转移类指令

助记符		功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟	提升效率
ACALL	addr11	绝对（短）调用子程序	2	24	6	4倍
LCALL	addr16	长调用子程序	3	24	6	4倍
RET		子程序返回	1	24	4	6倍
RETI		中断返回	1	24	4	6倍
AJMP	addr11	绝对（短）转移	2	24	3	8倍
LJMP	addr16	长转移	3	24	4	6倍
SJMP	re1	相对转移	2	24	3	8倍
JMP	@A+DPTR	相对于DPTR的间接转移	1	24	3	8倍
JZ	re1	累加器为零转移	2	24	3	8倍
JNZ	re1	累加器非零转移	2	24	3	8倍
CJNE	A，direct，re1	累加器与直接地址单元比较，不相等则转移	3	24	5	4.8倍
CJNE	A，#data，re1	累加器与立即数比较，不相等则转移	3	24	4	6倍
CJNE	Rn，#data，re1	寄存器与立即数比较，不相等则转移	3	24	4	6倍
CJNE	@Ri，#data，re1	间接RAM单元与立即数比较，不相等则转移	3	24	5	4.8倍
DJNZ	Rn，re1	寄存器减1，非零转移	3	24	4	6倍
DJNZ	direct，re1	直接地址单元减1，非零转移	3	24	5	4.8倍
NOP		空操作	1	12	1	12倍

布尔变量操作类指令

助记符		功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟	提升效率
CLR	C	清0进位位	1	12	1	12倍
CLR	bit	清0直接地址位	2	12	4	3倍
SETB	C	置1进位位	1	12	1	12倍
SETB	bit	置1直接地址位	2	12	4	3倍
CPL	C	进位位求反	1	12	1	12倍
CPL	bit	直接地址位求反	2	12	4	3倍
ANL	C，bit	进位位和直接地址位相“与”	2	24	3	8倍
ANL	C，/bit	进位位和直接地址位的反码相“与”	2	24	3	8倍
ORL	C，bit	进位位和直接地址位相“或”	2	24	3	8倍
ORL	C，/bit	进位位和直接地址位的反码相“或”	2	24	3	8倍
MOV	C，bit	直接地址位送入进位位	2	12	3	4倍
MOV	bit，C	进位位送入直接地址位	2	24	3	8倍
JC	re1	进位位为1则转移	2	24	3	8倍
JNC	re1	进位位为0则转移	2	24	3	8倍
JB	bit，re1	直接地址位为1则转移	3	24	4	6倍
JNB	bit，re1	直接地址位为0则转移	3	24	4	6倍
JBC	bit，re1	直接地址位为1则转移，该位清0	3	24	5	4.8倍

指令执行速度效率提升总结：

指令系统共包括 111 条指令，其中：

执行速度快 24 倍的	共 1 条
执行速度快 12 倍的	共 12 条
执行速度快 9.6 倍的	共 1 条
执行速度快 8 倍的	共 20 条
执行速度快 6 倍的	共 38 条
执行速度快 4.8 倍的	共 4 条
执行速度快 4 倍的	共 21 条
执行速度快 3 倍的	共 14 条

根据对指令的使用频率分析统计，STC12 系列 1T 的 8051 单片机比普通的 8051 单片机在同样的工作频率下运行速度提升了 8～12 倍。

指令执行时钟数统计（供参考）：

指令系统共包括 111 条指令，其中：

1 个时钟就可执行完成的指令	共 12 条
2 个时钟就可执行完成的指令	共 20 条
3 个时钟就可执行完成的指令	共 39 条
4 个时钟就可执行完成的指令	共 33 条
5 个时钟就可执行完成的指令	共 5 条
6 个时钟就可执行完成的指令	共 2 条

2.7 特殊功能寄存器映像 SFR Mapping

	Bit Addressable	Non Bit Addressable,不可位寻址(地址不能够被8整除的寄存器不能够位寻址)							
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8h		CH 0000,0000	CCAP0H 0000,0000	CCAP1H 0000,0000	CCAP2H 0000,0000	CCAP3H 0000,0000			FFh
F0h	B 0000,0000		PCA_PWM0 xxxx,xx00	PCA_PWM1 xxxx,xx00	PCA_PWM2 xxxx,xx00	PCA_PWM3 xxxx,xx00			F7h
E8h		CL 0000,0000	CCAP0L 0000,0000	CCAP1L 0000,0000	CCAP2L 0000,0000	CCAP3L 0000,0000			EFh
E0h	ACC 0000,0000	WDT_CONTR 0x00,0000	ISP_DATA 1111,1111	ISP_ADDRH 0000,0000	ISP_ADDRL 0000,0000	ISP_CMD xxxx,xx00	ISP_TRIG xxxx,xxxx	ISP_CONTR 0000,1000	E7h
D8h	CCON 00xx,0000	CMOD 0xxx,x000	CCAPM0 x000,0000	CCAPM1 x000,0000	CCAPM2 x000,0000	CCAPM3 x000,0000			DFh
D0h	PSW 0000,0000								D7h
C8h									CFh
C0h						ADC_CONTR 0000,0000	ADC_DATA 0000,0000	CLK_DIV xxxx,x000	C7h
B8h	IP x000,0000	SADEN don't use					ADC_LOW2 0000,0000		BFh
B0h	P3 1x11,1111	P3M0 0000,0000	P3M1 0000,0000					IPH x000,0000	B7h
A8h	IE 0000,0000	SADDR don't use							AFh
A0h	P2 1111,1111							TEST_WDT don't use	A7h
98h	SCON 0000,0000	SBUF xxxx,xxxx							9Fh
90h	P1 1111,1111	P1M0 0000,0000	P1M1 0000,0000	P0M0 0000,0000	P0M1 0000,0000	P2M0 0000,0000	P2M1 0000,0000		97h
88h	TCON 0000,0000	TMOD 0000,0000	TL0 0000,0000	TL1 0000,0000	TH0 0000,0000	TH1 0000,0000	AUXR 0000,00xx		8Fh
80h	P0 xxxx,1111	SP 0000,0111	DPL 0000,0000	DPH 0000,0000	SPSTAT 00xx,xxxx	SPCTL 0000,0100	SPDAT 0000,0000	PCON 0011,0000	87h
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

特别标出部分为在 Intel 8052 基础上新增加的特殊功能寄存器，一般用户可不管

STC12C5410AD 系列 8051 单片机内核特殊功能寄存器 C51 Core SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ACC	E0h	Accumulator									0000,0000
B	F0h	B Register									0000,0000
PSW	D0h	Program Status Word	CY	AC	F0	RS1	RS0	OV	F1	P	0000,0000
SP	81h	Stack Pointer									0000,0111
DPL	82h	Data Pointer Low Byte									0000,0000
DPH	83h	Data Pointer High Byte									0000,0000

STC12C5410AD 系列 8051 单片机系统管理特殊功能寄存器 System Management SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx
CLK_DIV	C7h	Clock Divder	-	-	-	-	-	CLKS2	CLKS1	CLKS0	xxxx,x000

STC12C5410AD 系列 8051 单片机 I/O 口 特殊功能寄存器 Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P0	80h	8-bit Port 0	-	-	-	-	P0.3	P0.2	P0.1	P0.0	xxxx,1111
P0M0	93h										0000,0000
P0M1	94h										0000,0000
P1	90h	8-bit Port 1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	1111,1111
P1M0	91h										0000,0000
P1M1	92h										0000,0000
P2	A0h	8-bit Port 2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	1111,1111
P2M0	95h										0000,0000
P2M1	96h										0000,0000
P3	B0h	8-bit Port 3	P3.7	-	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	1x11,1111
P3M0	B1h										0000,0000
P3M1	B2h										0000,0000

STC12C5410AD 系列 8051 单片机 定时器 特殊功能寄存器 Timer SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
TMOD	89h	Timer / Counter 0 and 1 Modes	GATE GATE1	C/T# C/T1#	M1 M1_1	M0 M1_0	GATE GATE0	C/T# C/T0#	M1 M0_1	M0 M0_0	0000,0000
TL0	8Ah	Timer / Counter 0 Low Byte									0000,0000
TH0	8Ch	Timer / Counter 0 High Byte									0000,0000
TL1	8Bh	Timer / Counter 1 Low Byte									0000,0000
TH1	8Dh	Timer / Counter 1 High Byte									0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx

STC12C5410AD 系列 8051 单片机 串行口 特殊功能寄存器 Serial I/O Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
SBUF	99h	Serial Data Buffer									xxxx,xxxx
SADEN	B9h	Slave Address Mask									0000,0000
SADDR	A9h	Slave Address									0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx

STC12C5410AD 系列 8051 单片机 看门狗定时器 特殊功能寄存器 Watch Dog Timer SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
WDT_CONTR	E1h	Watch-Dog-Timer Control register	WDT_FLAG	-	EN_WDT	CLR_WDT	IDLE_WDT	PS2	PS1	PS0	xx00,0000

STC12C5410AD 系列 1T 8051 单片机 中断 特殊功能寄存器 Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	EPCA_LVD	EADC_SPI	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	PPCA_LVD	PADC_SPI	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	-	PPCA_LVDH	PADC_SPIH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx
ADC_CONTR	C5h	A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0xx0,0000
SPSTAT	84h	SPI Status Register	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
CCON	D8h	PCA Control Register	CF	CR	-	-	CCF3	CCF2	CCF1	CCF0	00xx,0000
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CCAPM2	DCh	PCA Module 2 Mode Register	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	x000,0000
CCAPM3	DDh	PCA Module 3 Mode Register	-	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	x000,0000

STC12C5410AD 系列 8051 单片机 PCA/PWM 特殊功能寄存器 PCA/PWM SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CCON	D8h	PCA Control Register	CF	CR	-	-	CCF3	CCF2	CCF1	CCF0	00xx,0000
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CCAPM2	DCh	PCA Module 2 Mode Register	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	x000,0000
CCAPM3	DDh	PCA Module 3 Mode Register	-	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	x000,0000
CL	E9h	PCA Base Timer Low									0000,0000
CH	F9h	PCA Base Timer High									0000,0000
CCAP0L	EAh	PCA Module-0 Capture Register Low									0000,0000
CCAP0H	FAh	PCA Module-0 Capture Register High									0000,0000
CCAP1L	EBh	PCA Module-1 Capture Register Low									0000,0000
CCAP1H	FBh	PCA Module-1 Capture Register High									0000,0000
CCAP2L	ECh	PCA Module-2 Capture Register Low									0000,0000
CCAP2H	FCh	PCA Module-2 Capture Register High									0000,0000
CCAP3L	EDh	PCA Module-3 Capture Register Low									0000,0000
CCAP3H	FDh	PCA Module-3 Capture Register High									0000,0000
PCA_PWM0	F2h	PCA PWM Mode Auxiliary Register 0	-	-	-	-	-	-	EPC0H	EPC0L	xxxx,xx00
PCA_PWM1	F3h	PCA PWM Mode Auxiliary Register 1	-	-	-	-	-	-	EPC1H	EPC1L	xxxx,xx00
PCA_PWM2	F4h	PCA PWM Mode Auxiliary Register 2	-	-	-	-	-	-	EPC2H	EPC2L	xxxx,xx00
PCA_PWM3	F5h	PCA PWM Mode Auxiliary Register 3	-	-	-	-	-	-	EPC3H	EPC3L	xxxx,xx00

STC12C5410AD 系列 8051 单片机 ISP/IAP 特殊功能寄存器 ISP/IAP SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ISP_DATA	E2h	ISP/IAP Flash Data Register									1111,1111
ISP_ADDRH	E3h	ISP/IAP Flash Address High									0000,0000
ISP_ADDRL	E4h	ISP/IAP Flash Address Low									0000,0000
ISP_CMD	E5h	ISP/IAP Flash Command Register	-	-	-	-	-	-	MS1	MS0	xxxx,x000
ISP_TRIG	E6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
ISP_CONTR	E7h	ISP/IAP Control Register	ISPEN	SWBS	SWRST	CMD_FAIL	1	WT2	WT1	WT0	0000,1000

2.8 中断优先级及中断寄存器

2.8.1 中断优先级

STC12C5410AD 及 STC12C2052AD 系列单片机 中断优先级及中断查询次序，与 8051 完全兼容

Interrupt Source 中断源	Vector Address 中断向量地址	Polling Sequence 中断查询次序	中断优先级设置 (IPH,IP)	优先级0 最低	优先级1	优先级2	优先级3 最高	Interrupt Request 中断请求标志位	Interrupt Enable Control Bit 中断允许控制位
/INT0	0003H	0(最优先)	PX0H,PX0	0,0	0,1	1,0	1,1	IE0	EX0 / EA
Timer 0	000BH	1	PT0H,PT0	0,0	0,1	1,0	1,1	TF0	ET0 / EA
/INT1	0013H	2	PX1H,PX1	0,0	0,1	1,0	1,1	IE1	EX1 / EA
Timer 1	001BH	3	PT1H,PT1	0,0	0,1	1,0	1,1	TF1	ET1 / EA
UART	0023H	4	PSH, PS	0,0	0,1	1,0	1,1	RI + TI	ES / EA
ADC/SPI	002BH	5	PADC_SPIH,PADC_SPI	0,0	0,1	1,0	1,1	ADC_FLAG + SPIF	(EADC+ESPI)/EADC_SPI/EA
PCA/LVD	0033H	6	PPCA_LVDH,PPCA_LVD	0,0	0,1	1,0	1,1	CF + CCF0 + CCF1 + CCF2 + CCF3+ LVDF	(ECF+ECCF0+ECCF1+ECCF2+ ECCF3+ELVD)/EPCA_LVD/EA

通过设置新增加的特殊功能寄存器 IPH 中的相应位，可将中断优先级设为四级，如果只设置 IP，那么中断优先级就只有两级，与传统 8051 单片机两级中断优先级完全兼容。

如果使用 C 语言编程，中断查询次序号就是中断号，例如：

```
void  Int0_Routine(void) interrupt 0;
void  UART_Routine(void) interrupt 4;
void  ADC_SPI_Routine(void) interrupt 5;
void  PCA_LVD_Routine(void) interrupt 6;
```

STC12C5410AD 及 STC12C2052AD 系列 1T 8051 单片机 中断 特殊功能寄存器 Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	EPCA_LVD	EADC_SPI	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	PPCA_LVD	PADC_SPI	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	-	PPCA_LVDH	PADC_SPIH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx
ADC_CONTR	C5h	A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0xx0,0000
SPSTAT	84h	SPI Status Register	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
CCON	D8h	PCA Control Register	CF	CR	-	-	CCF3	CCF2	CCF1	CCF0	00xx,0000
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CCAPM2	DCh	PCA Module 2 Mode Register	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	x000,0000
CCAPM3	DDh	PCA Module 3 Mode Register	-	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	x000,0000

PCA/PWM 特殊功能寄存器 ，其中部分位与 PCA 中断有关

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CCON	D8h	PCA Control Register	CF	CR	-	-	CCF3	CCF2	CCF1	CCF0	00xx,0000
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CCAPM2	DCh	PCA Module 2 Mode Register	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	x000,0000
CCAPM3	DDh	PCA Module 3 Mode Register	-	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	x000,0000
CL	E9h	PCA Base Timer Low									0000,0000
CH	F9h	PCA Base Timer High									0000,0000
CCAP0L	EAh	PCA Module-0 Capture Register Low									0000,0000
CCAP0H	FAh	PCA Module-0 Capture Register High									0000,0000
CCAP1L	EBh	PCA Module-1 Capture Register Low									0000,0000
CCAP1H	FBh	PCA Module-1 Capture Register High									0000,0000
CCAP2L	ECh	PCA Module-2 Capture Register Low									0000,0000
CCAP2H	FCh	PCA Module-2 Capture Register High									0000,0000
CCAP3L	EDh	PCA Module-3 Capture Register Low									0000,0000
CCAP3H	FDh	PCA Module-3 Capture Register High									0000,0000
PCA_PWM0	F2h	PCA PWM Mode Auxiliary Register 0	-	-	-	-	-	-	EPC0H	EPC0L	xxxx,xx00
PCA_PWM1	F3h	PCA PWM Mode Auxiliary Register 1	-	-	-	-	-	-	EPC1H	EPC1L	xxxx,xx00
PCA_PWM2	F4h	PCA PWM Mode Auxiliary Register 2	-	-	-	-	-	-	EPC2H	EPC2L	xxxx,xx00
PCA_PWM3	F5h	PCA PWM Mode Auxiliary Register 3	-	-	-	-	-	-	EPC3H	EPC3L	xxxx,xx00

STC12C5410AD 系列 8051 单片机 SPI 功能模块特殊功能寄存器 其中 SPIF 位与中断有关

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
SPCTL	85h	SPI Control Register	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	0000,0000
SPSTAT	84h	SPI Status Register	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
SPDAT	86h	SPI Data Register									0000,0000

2.8.2 几个新增加的中断控制位

如果要允许A/D转换中断则需要将几个相应的控制位置1:

1、将EADCI置1，允许ADC中断，这是ADC中断的单独控制位。

2、将EADCI_SPI置1，允许ADC中断及SPI中断，这是ADC中断及SPI中断的总中断控制位，此位不打开，也是无法产生ADC中断的。

3、将EA置1，打开单片机总中断控制位，此位不打开，也是无法产生ADC中断的

A/D中断服务程序中要用软件清A/D中断请求标志位ADC_FLAG。

如果要允许SPI中断则需要将几个相应的控制位置1:

1、将ESPI置1，允许SPI中断，这是SPI中断的单独控制位。

2、将EADCI_SPI置1，允许ADC中断及SPI中断，这是ADC中断及SPI中断的总中断控制位，此位不打开，也是无法产生SPI中断的。

3、将EA置1，打开单片机总中断控制位，此位不打开，也是无法产生SPI中断的

SPI中断服务程序中要用软件清SPI中断请求标志位SPIF。

如果要允许低压中断则需要将几个相应的控制位置1:

1、将ELVDI置1，允许低压检测中断，这是低压中断的单独控制位。

2、将EPCA_LVD置1，允许PCA模块中断及低压检测中断，这是PCA模块中断及低压检测中断的总中断控制位，此位不打开，也是无法产生低压检测中断的。

3、将EA置1，打开单片机总中断控制位，此位不打开，也是无法产生低压检测中断的

低压检测中断服务程序中要用软件清低压中断请求标志位LVDF。

5V单片机，3.7V（±0.1）以下为低压，3V 单片机，2.4V（±0.1）以下为低压，

如ELVDI=1（允许低压中断），则会产生低压中断

STC12C5410AD系列单片机现版本（A/B/C版）无低压检测中断，现是低压复位

---5V单片机3.7V以下复位，3V单片机2.4V以下复位。

只有STC12C2052AD系列单片机有低压检测中断,现STC12C5410AD系列单片机无低压检测中断.

如果要允许PCA中断则需要将几个相应的控制位置1:

1、将ECF/ECCF0/ECCF1/ECCF2/ECCF3中断允许位需要置1的位置1，允许PCA模块中相应的模块产生中断，这些是PCA模块中相应模块的单独控制位。

2、将EPCA_LVD置1，允许PCA模块中断及低压检测中断，这是PCA模块中断及低压检测中断的总中断控制位，此位不打开，也是无法产生PCA中断的。

3、将EA置1，打开单片机总中断控制位，此位不打开，也是无法产生PCA中断的

PCA中断服务程序中要用软件清相应的PCA中断请求标志位CF/CCF0/CCF1/CCF2/CCF3。

STC12C2052AD系列因为只有两路PCA，所以没有ECCF2/ECCF3,没有CCF2/CCF3

2.9 定时器 0/ 定时器 1，UART 串口的速度

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx

定时器 0 和定时器 1：

STC12C5410AD 和 STC12C2052AD 系列是 1T 的 8051 单片机，为了兼容传统 8051，定时器 0 和定时器 1 复位后是传统 8051 的速度，即 12 分频，这是为了兼容传统 8051。但也可不进行 12 分频，实现真正的 1T。

T0x12： 0，定时器 0 是传统 8051 速度，12 分频；1，定时器 0 的速度是传统 8051 的 12 倍，不分频

T1x12： 0，定时器 1 是传统 8051 速度，12 分频；1，定时器 1 的速度是传统 8051 的 12 倍，不分频

如果 UART 串口用定时器 1 做波特率发生器，T1x12 位就可以控制 UART 串口是 12T 还是 1T 了。

UART 串口的模式 0：

STC12C5410AD 和 STC12C2052AD 系列是 1T 的 8051 单片机，为了兼容传统 8051，UART 串口复位后是兼容传统 8051 的。

UART_M0x6： 0，UART 串口的模式 0 是传统 12T 的 8051 速度，12 分频；

1，UART 串口的模式 0 的速度是传统 12T 的 8051 的 6 倍，2 分频

如果用定时器 T1 做波特率发生器时,UART 串口的速度由 T1 的溢出率决定

EADCI： 0，禁止 A/D 中断； 1，允许 A/D 中断

ESPI： 0，禁止 SPI 中断； 1，允许 SPI 中断

ELVDI： 0，禁止低压中断； 1，允许低压中断

5V 单片机，3.7V 以下为低压，3V 单片机，2.4V 以下为低压，

如 ELVDI=1（允许低压中断），则会产生低压中断，现版本无低压检测中断，是低压复位。

STC12C5410AD 系列无低压检测中断,只有 STC12C2052AD 系列单片机才有低压检测中断。

2.10 STC12 系列单片机内部 / 外部工作时钟可选

STC12C5410AD 及 STC12C2052AD 系列是 1T 的 8051 单片机，系统时钟兼容传统 8051。

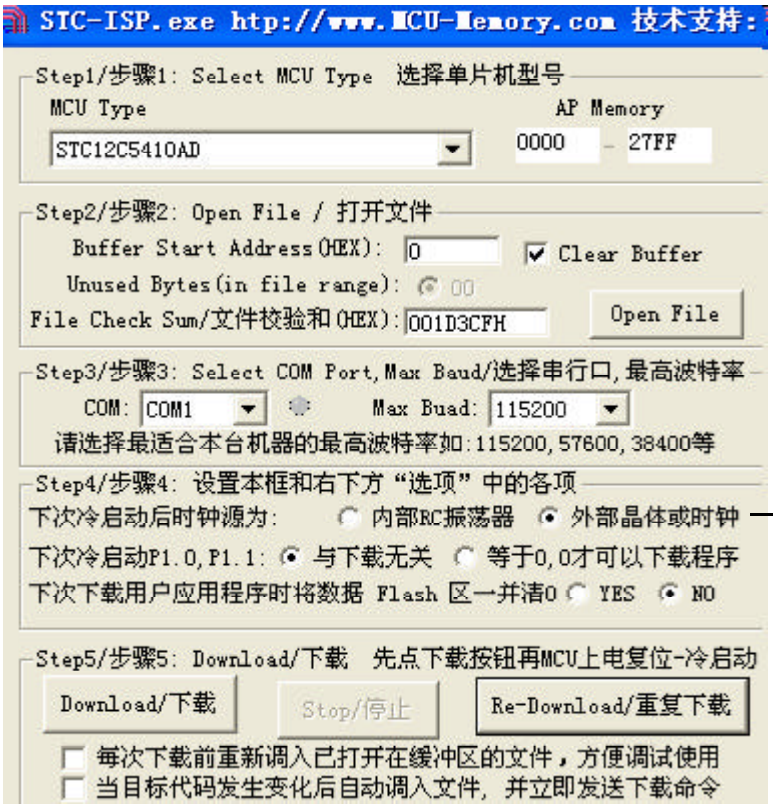
现出厂标准配置是使用芯片内部的 R/C 振荡器，5V 单片机常温下频率是 5MHz - 6.9MHz,因为随着温度的变化，内部 R/C 振荡器的频率会有一些温飘，再加上制造误差，应认为是 4MHz - 8MHz。故内部 R/C 振荡器只适用于对时钟频率要求不敏感的场合。

在对 STC12C5410AD 或 STC12C2052AD 系列单片机进行 ISP 下载用户程序时，可以在选项中选择：

“ 下次冷启动后时钟源为外部晶体或时钟 ”

这样下载完用户程序后，停电，再冷启动后单片机的工作时钟使用的就不是内部 R/C 振荡器，而是外部晶体振荡后产生的高精度时钟了（接在 XTAL1/XTAL2 管脚上），也可以直接从 XTAL1 脚输入外部时钟，XTAL2 脚浮空。用户以后外部必须接晶体或时钟单片机才可以工作。

如果已被设置成用外部晶体或时钟工作的单片机，还要再设回使用内部 R/C 振荡器工作，则需给单片机外接晶体或时钟，再对 STC12C5410AD 或 STC12C2052AD 系列单片机进行 ISP 下载用户程序时在选项中选择：



选择下次冷启动后时钟源为：
1.内部 R/C 振荡器
2.外部晶体或时钟

下载用户程序成功后，新的设置就设置进单片机内部了，但必须停电后再上电单片机才会用新的设置工作

2.11 时钟分频及分频寄存器

时钟分频寄存器，可将时钟分成较低频率工作

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CLK_DIV	C7h	Clock Divder	-	-	-	-	-	CLKS2	CLKS1	CLKS0	xxxx,x000

如用户系统希望大幅降低功耗，还可对系统时钟进行分频。（STC12C2052AD 系列不要使用分频）

CLKS2	CLKS1	CLKS0	分频后CPU的实际工作时钟
0	0	0	系统时钟(外部时钟或内部R/C振荡时钟)
0	0	1	系统时钟/2
0	1	0	系统时钟/4
0	1	1	系统时钟/8
1	0	0	系统时钟/16
1	0	1	系统时钟/32
1	1	0	系统时钟/64
1	1	1	系统时钟/128

STC12C5410AD 系列单片机可以在空闲模式时分频工作,也可以在正常工作时分频

第三章 STC12 系列单片机的 I/O 口结构

3.1 I/O 口各种不同的工作模式及配置介绍

I/O 口配置

STC12C5410AD 系列单片机其所有 I/O 口均可由软件配置成 4 种工作类型之一，如下表所示。4 种类型分别为：准双向口（标准 8051 输出模式）、推挽输出、仅为输入（高阻）或开漏输出功能。每个口由 2 个控制寄存器中的相应位控制每个引脚工作类型。STC12C5410AD 系列单片机上电复位后为准双向口（传统 8051 的 I/O 口）模式。2V 以上时为高电平，0.8V 以下时为低电平。

I/O 口工作类型设定

P3 口设定 <P3.7, x ,P3.5,P3.4,P3.3,P3.2,P3.1,P3.0 无 P3.6 口>

P3M0【7：0】	P3M1【7：0】	I/O 口模式
0	0	准双向口(传统 8051 I/O 口模式）， 灌电流可达 20mA，拉电流为 230μA， 由于制造误差，实际为 250uA～150uA
0	1	推挽输出（强上拉输出，可达 20mA,要加限流电阻）
1	0	仅为输入（高阻）
1	1	开漏(Open Drain)，内部上拉电阻断开，要外加

P2 口设定 <P2.7,P2.6,P2.5,P2.4,P2.3,P2.2,P2.1,P2.0>

P2M0【7：0】	P2M1【7：0】	I/O 口模式
0	0	准双向口(传统 8051 I/O 口模式）， 灌电流可达 20mA，拉电流为 230μA， 由于制造误差，实际为 250uA～150uA
0	1	推挽输出（强上拉输出，可达 20mA，要加限流电阻）
1	0	仅为输入（高阻）
1	1	开漏(Open Drain)，内部上拉电阻断开，要外加

P1 口设定 <P1.7,P1.6,P1.5,P1.4,P1.3,P1.2,P1.1,P1.0>

P1M0【7：0】	P1M1【7：0】	I/O 口模式（P1.x 如做 A/D 使用，需先将其设置成开漏或高阻输入）
0	0	准双向口（传统 8051 I/O 口模式）， 灌电流可达 20mA，拉电流为 230μA， 由于制造误差，实际为 250uA～150uA
0	1	推挽输出（强上拉输出，可达 20mA，要加限流电阻）
1	0	仅为输入（高阻），如果该 I/O 口需作为 A/D 使用，可选此模式
1	1	开漏(Open Drain)，如果该 I/O 口需作为 A/D 使用，可选此模式

P0 口设定 < x , x , x , x , P0.3, P0.2, P0.1, P0.0 无 P0.7,P0.6,P0.5,P0.4 口>

P0M0【7：0】	P0M1【7：0】	I/O 口模式
0	0	准双向口(传统 8051 I/O 口模式）， 灌电流可达 20mA，拉电流为 230μA， 由于制造误差，实际为 250uA～150uA
0	1	推挽输出（强上拉输出，可达 20mA，要加限流电阻）
1	0	仅为输入（高阻）
1	1	开漏(Open Drain)，内部上拉电阻断开，要外加

```
举例:    MOV    P1M0,  #11000000B
         MOV    P1M1,  #10100000B
    ;P1.7 为开漏,P1.6 为高阻输入,P1.5 为强推挽输出,P1.4/P1.3/P1.2/P1.1/P1.0 为弱上拉
```

注意:

虽然每个 I/O 口在弱上拉时都能承受 20mA 的灌电流(还是要加限流电阻,如 1K，560 Ω等),在强推挽输出时都能输出 20mA 的拉电流（也要加限流电阻),但整个芯片的工作电流推荐不要超过 55mA。即从 MCU-VCC 流入的电流不超过 55mA,从 MCU-Gnd 流出电流不超过 55mA,整体流入 / 流出电流都不能超过 55mA.

3.2 I/O口各种不同的工作模式结构框图

1.准双向口输出配置

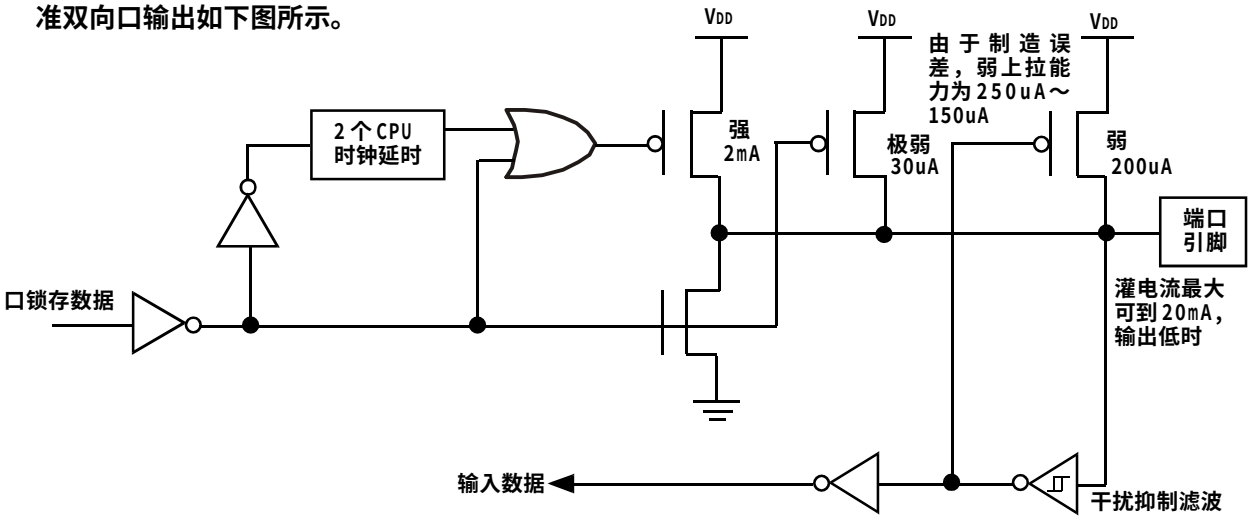
准双向口输出类型可用作输出和输入功能而不需重新配置口线输出状态。这是因为当口线输出为1时驱动能力很弱，允许外部装置将其拉低。当引脚输出为低时，它的驱动能力很强，可吸收相当大的电流。准双向口有3个上拉晶体管适应不同的需要。

在3个上拉晶体管中，有1个上拉晶体管称为“弱上拉”，当口线寄存器为1且引脚本身也为1时打开。此上拉提供基本驱动电流使准双向口输出为1。如果一个引脚输出为1而由外部装置下拉到低时，弱上拉关闭而“极弱上拉”维持开状态，为了把这个引脚强拉为低，外部装置必须有足够的灌电流能力使引脚上的电压降到门槛电压以下。

第2个上拉晶体管，称为“极弱上拉”，当口线锁存为1时打开。当引脚悬空时，这个极弱的上拉源产生很弱的上拉电流将引脚上拉为高电平。

第3个上拉晶体管称为“强上拉”。当口线锁存器由0到1跳变时，这个上拉用来加快准双向口由逻辑0到逻辑1转换。当发生这种情况时，强上拉打开约2个时钟以使引脚能够迅速地上拉到高电平。

准双向口输出如下图所示。



STC12LE5410系列单片机为3V器件，如果用户在引脚加上5V电压，将会有电流从引脚流向VDD，这样导致额外的功率消耗。因此，建议不要在准双向口模式中向3V单片机引脚施加5V电压,如使用的话，要加限流电阻，或用二极管做输入隔离，或用三极管做输出隔离。

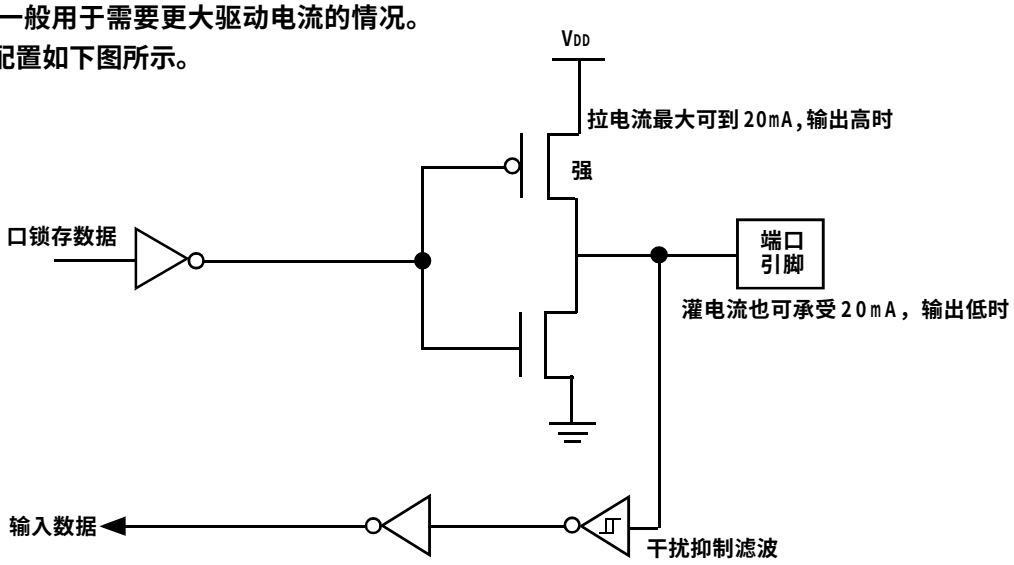
准双向口带有一个施密特触发输入以及一个干扰抑制电路。

准双向口读外部状态前,要先锁存为 ‘1’,才可读到外部正确的状态.

2.推挽输出配置

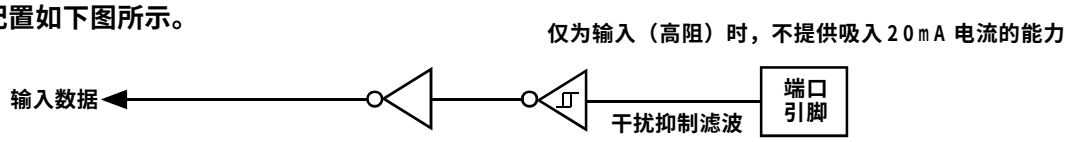
推挽输出配置的下拉结构与开漏输出以及准双向口的下拉结构相同，但当锁存器为1时提供持续的强上拉。推挽模式一般用于需要更大驱动电流的情况。

推挽引脚配置如下图所示。



3．仅为输入（高阻）配置

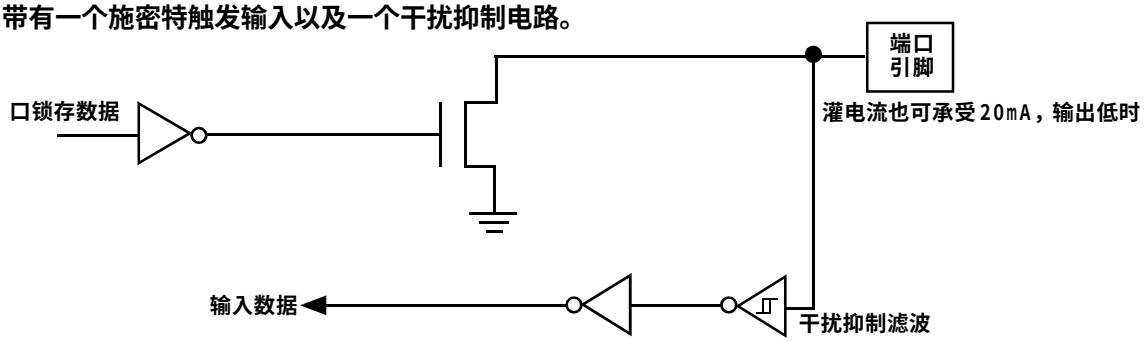
输入口配置如下图所示。



输入口带有一个施密特触发输入以及一个干扰抑制电路。

4.开漏输出配置

当口线锁存器为0时，开漏输出关闭所有上拉晶体管。当作为一个逻辑输出时，这种配置方式必须有外部上拉，一般通过电阻外接到VDD。这种方式的下拉与准双向口相同。输出口线配置如下图所示。
开漏端口带有一个施密特触发输入以及一个干扰抑制电路。



关于I/O口应用注意事项：

少数用户反映I/O口有损坏现象,后发现是

有些是I/O口由低变高读外部状态时,读不对,实际没有损坏,软件处理一下即可

是因为1T的8051单片机速度太快了,软件执行由低变高指令后立即读外部状态,此时由于实际输出还没有变高,就有可能读不对,正确的方法是在软件设置由低变高后加1到2个空操作指令延时,再读就对了.

有些实际没有损坏,加上拉电阻就OK了

是因为外围接的是SPI/I2C等漏极开漏的电路,要加10K上拉电阻.

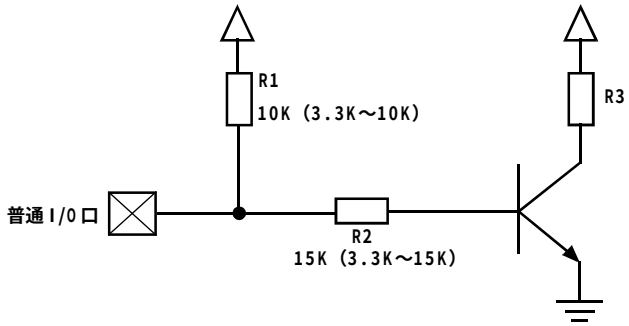
有些是外围接的是NPN三极管,没有加上拉电阻,其实基极串多大电阻,I/O口就应该上拉多大的电阻,或者将该I/O口设置为强推挽输出.

有些确实是损坏了,原因:

发现有些是驱动LED发光二极管没有加限流电阻,建议加1K以上的限流电阻,至少也要加470欧姆以上

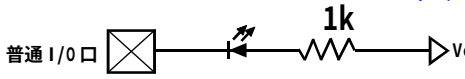
发现有些是做行列矩阵按键扫描电路时,实际工作时没有加限流电阻,实际工作时可能出现2个I/O口均输出为低,并且在按键按下时,短接在一起,我们知道一个CMOS电路的2个输出脚不应该直接短接在一起,按键扫描电路中,此时一个口为了读另外一个口的状态,必须先置高才能读另外一个口的状态,而8051单片机的弱上拉口在由0变为1时,会有2个时钟的强推挽高输出电流,输出到另外一个输出为低的I/O口,就有可能造成I/O口损坏.建议在其中的一侧加1K限流电阻,或者在软件处理上,不要出现按键两端的I/O口同时为低.

3.3 一种典型三极管控制电路



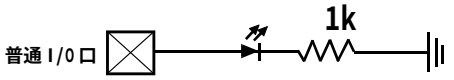
如果用弱上拉控制，建议加上拉电阻 R1（3.3K～10K），如果不加上拉电阻 R1（3.3K～10K），建议 R2 的值在 15K 以上，或用强推挽输出。

3.4 典型发光二极管控制电路



弱上拉 / 准双向口，用灌电流驱动发光二极管

限流电阻尽量大于 1K，最小不要小于 470 欧姆

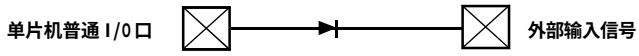


推挽 / 强上拉口，用拉电流驱动发光二极管

3.5 3V/5V 混合系统 I/O 口互连

STC12C5410AD 系列 5V 单片机连接 3V 器件时，为防止 3V 器件承受不了 5V，可将相应的 I/O 口设置成开漏配置，断开内部上拉电阻，相应的 I/O 口外部加 10K 上拉电阻到 3V 器件的 Vcc，这样高电平是 3V，低电平是 0V，输入输出一切正常。

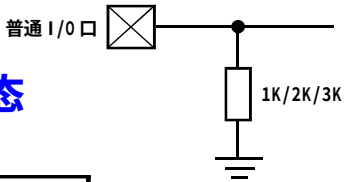
STC12LE5410AD 系列 3V 单片机连接 5V 器件时，为防止 3V 器件承受不了 5V，如果相应的 I/O 口是输入，可在该 I/O 口上串接一个隔离二极管，隔离高压部分。外部信号电压高于单片机工作电压时截止，I/O 口此时已内部上拉到高电平；外部信号电压为低时导通，I/O 口被钳位在 0.7V，小于 0.8V 时单片机就认为是低电平。



3.6 如何让 I/O 口上电复位时为低电平

普通8051单片机上电复位时普通I/O口为弱上拉高电平输出,而很多实际应用要求上电时某些I/O口为低电平输出,否则所控制的系统(如马达)就会误动作,现STC12系列单片机由于既有弱上拉输出又有强推挽输出,就可以很轻松的解决此问题。

现可在 STC12 系列单片机 I/O 口上加一个下拉电阻(1K/2K/3K),这样上电复位时，虽然单片机内部 I/O 口是弱上拉 / 高电平输出，但由于内部上拉能力有限，而外部下拉电阻又较小,无法将其拉高,所以该 I/O 口上电复位时外部为低电平。如果要将此 I/O 口驱动为高电平，可将此 I/O 口设置为强推挽输出，而强推挽输出时，I/O 口驱动电流可达 20mA,故肯定可以将该口驱动为高电平输出。

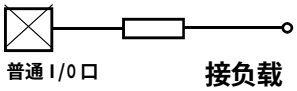


3.7 PWM 输出时 I/O 口的状态

当某个 I/O 口作为 PWM 输出用时,该口的状态:

PWM之前口的状态	PWM时口的状态
弱上拉/准双向口	强推挽输出 / 强上拉输出 要加输出限流电阻10K-1K
强推挽输出	强推挽输出 / 强上拉输出 要加输出限流电阻10K-1K
仅为输入/高阻	PWM无效
开漏	开漏

限流电阻用 10K 到 1K



第四章 STC12 系列单片机看门狗应用及软件复位

4.1 看门狗应用及测试程序

4.1.1 看门狗应用介绍

适用型号： STC12C5410AD 系列,STC12C2052AD 系列

在工业控制 / 汽车电子 / 航空航天等需要高可靠性的系统中,为了防止“系统在异常情况下，受到干扰，MCU/CPU 程序跑飞，导致系统长时间异常工作”,通常是引进看门狗,如果 MCU/CPU 不在规定的时间内按要求访问看门狗,就认为MCU/CPU处于异常状态,看门狗就会强迫MCU/CPU复位,使系统重新从头开始按规律执行用户程序。STC12 系列单片机内部也引进了此看门狗功能，使单片机系统可靠性设计变得更加方便 / 简洁。为此功能，我们增加如下特殊功能寄存器 WDT_CONTR：

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
WDT_CONTR	E1h	Watch-Dog-Timer Control register	WDT_FLAG	-	EN_WDT	CLR_WDT	IDLE_WDT	PS2	PS1	PS0	xx00,0000

Symbol 符号 Function 功能

WDT_FLAG When WDT overflows， this bit is set. It can be cleared by software.
看门狗溢出标志位,当溢出时，该位由硬件置 1，可用软件将其清 0。

EN_WDT Enable WDT bit. When set, WDT is started
看门狗允许位， 当设置为“ 1 ”时，看门狗启动。

CLR_WDT WDT clear bit. When set, WDT will recount. Hardware will automatically clear this bit.
看门狗清“ 0 ”位，当设为“ 1 ”时，看门狗将重新计数。硬件将自动清“ 0 ”此位。

IDLE_WDT When set, WDT is enabled in IDLE mode. When clear, WDT is disabled in IDLE
看门狗“ IDLE ”模式位,当设置为“ 1 ”时,看门狗定时器在“ 空闲模式 ”计数
当清“ 0 ”该位时，看门狗定时器在“ 空闲模式 ”时不计数

PS2, PS1, PS0 Pre-scale value of Watchdog timer is shown as the bellowed table:
看门狗定时器预分频值，如下表所示

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @20MHz
0	0	0	2	39.3 mS
0	0	1	4	78.6 mS
0	1	0	8	157.3 mS
0	1	1	16	314.6 mS
1	0	0	32	629.1 mS
1	0	1	64	1.25S
1	1	0	128	2.5S
1	1	1	256	5S

The WDT period is determined by the following equation 看门狗溢出时间计算
看门狗溢出时间 = (12 x Pre-scale x 32768) / Oscillator frequency
设时钟为 12MHz：
看门狗溢出时间 = (12 x Pre-scale x 32768) / 12000000 = Pre-scale x 393216 / 12000000

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @12MHz
0	0	0	2	65.5 mS
0	0	1	4	131.0 mS
0	1	0	8	262.1 mS
0	1	1	16	524.2 mS
1	0	0	32	1.0485S
1	0	1	64	2.0971S
1	1	0	128	4.1943S
1	1	1	256	8.3886S

设时钟为 11.0592MHz：

看门狗溢出时间 = (12 x Pre-scale x 32768) / 11059200 = Pre-scale x 393216 / 11059200

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @11.0592MHz
0	0	0	2	71.1 mS
0	0	1	4	142.2 mS
0	1	0	8	284.4 mS
0	1	1	16	568.8 mS
1	0	0	32	1.1377S
1	0	1	64	2.2755S
1	1	0	128	4.5511S
1	1	1	256	9.1022S

汇编语言程序示例

```
WDT_CONTR   DATA    0E1H ;    或者   WDT_CONTR  EQU   0E1H
;复位入口
    ORG     0000H
    LJMP    Initial
    …
    ORG     0060H
Initial:
    MOV     WDT_CONTR,  #00111100B; Load initial value 看门狗定时器控制寄存器初始化
                ; EN_WDT = 1, CLR_WDT = 1, IDLE_WDT = 1, PS2 = 1, PS1 = 0, PS0 = 0
    …
Main_Loop:
    LCALL   Display_Loop
    LCALL   Keyboard_Loop
    …
    MOV     WDT_CONTR,  #00111100B ;  喂狗，  不要用 ORL    WDT_CONTR,  #00010000B
    …
    LJMP    Main_Loop
```

C语言程序示例

```
#include<reg52.h>
sfr     WDT_CONTR   =   0xe1;
void main()
{   …
    WDT_CONTR   =   0x3c;
       /* 0011,1100 EN_WDT = 1,CLR_WDT = 1,IDLE_WDT = 1,PS2 = 1,PS1 = 0,PS0 = 0 */
    while(1){
        display();
        keyboard();
        …
        WDT_CONTR   =   0x3c;  /* 喂狗，  不要用  WDT_CONTR   =   WDT_CONTR | 0x10;*/
    }
}
```

4.1.2 一个完整的看门狗测试程序，在宏晶的下载板上可以直接测试

本程序验证 STC12C5410AD 系列及 STC12C2052AD 系列单片机的看门狗及其溢出时间计算公式

```
;/* ---  STC International Limited ------------------------- */
;/* ---  宏晶科技 姚永平  设计 2006/1/6   V1.0 -------------- */
;/* ---  演示 STC12C5410AD 系列 MCU 看门狗及其溢出时间计算公式 - */
;/* ---  演示 STC12C2052AD 系列 MCU 看门狗及其溢出时间计算公式 - */
;/* ---  Mobile: 13922805190 ------------------------------- */
;/* ---  Fax: 0755-82944243 -------------------------------- */
;/* ---  Tel: 0755-82948409 -------------------------------- */
;/* ---  Web: www.mcu-memory.com --------------------------- */
;如果要在程序中使用或在文章中引用该程序,请在程序中或文章中注明使用了宏晶科技的资料及程序
;本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过,相关的工作状态在 P1 口上显示
;看门狗及其溢出时间 = (12 * Pre_scale *32768)/Oscillator frequency
WDT_CONTR       EQU 0E1H ;看门狗地址
WDT_TIME_LED    EQU P1.5 ;用 P1.5 控制看门狗溢出时间指示灯,
                         ;看门狗溢出时间可由该指示灯亮的时间长度或熄灭的时间长度表示
WDT_FLAG_LED    EQU P1.7 ;用 P1.7 控制看门狗溢出复位指示灯，如点亮表示为看门狗溢出复位
Last_WDT_Time_LED_Status  EQU 00H ;位变量，存储看门狗溢出时间指示灯的上一次状态位
;WDT 复位时间(所用的 Oscillator frequency = 18.432MHz):
;Pre_scale_Word  EQU 00111100B   ;清 0,启动看门狗，预分频数 =32,   0.68S
Pre_scale_Word  EQU 00111101B    ;清 0,启动看门狗，预分频数 =64,   1.36S
;Pre_scale_Word  EQU 00111110B   ;清 0,启动看门狗，预分频数 =128,  2.72S
;Pre_scale_Word  EQU 00111111B   ;清 0,启动看门狗，预分频数 =256,  5.44S
    ORG   0000H
    AJMP  MAIN
    ORG   0100H
MAIN:
    MOV   A, WDT_CONTR          ;检测是否为看门狗复位
    ANL   A, #10000000B
    JNZ   WDT_Reset             ;WDT_CONTR.7 = 1，看门狗复位，跳转到看门狗复位程序
;WDT_CONTR.7 = 0,上电复位，冷启动，RAM 单元内容为随机值
    SETB  Last_WDT_Time_LED_Status       ;上电复位,
                                         ;初始化看门狗溢出时间指示灯的状态位 = 1
    CLR   WDT_TIME_LED                   ;上电复位，点亮看门狗溢出时间指示灯
    MOV   WDT_CONTR, #Pre_scale_Word     ;启动看门狗
WAIT1:
    SJMP  WAIT1                  ;循环执行本语句(停机)，等待看门狗溢出复位

;WDT_CONTR.7 = 1,看门狗复位，热启动，RAM 单元内容不变，为复位前的值
WDT_Reset:                       ;看门狗复位，热启动
    CLR   WDT_FLAG_LED           ;是看门狗复位,点亮看门狗溢出复位指示灯

    JB    Last_WDT_Time_LED_Status, Power_Off_WDT_TIME_LED;为 1 熄灭相应的灯,为 0 亮相应灯
    ;根据看门狗溢出时间指示灯的上一次状态位设置 WDT_TIME_LED 灯,
    ;若上次亮本次就熄灭，若上次熄灭本次就亮
```

```
    CLR   WDT_TIME_LED              ;上次熄灭本次点亮看门狗溢出时间指示灯
    CPL   Last_WDT_Time_LED_Status ;将看门狗溢出时间指示灯的上一次状态位取反
WAIT2:
    SJMP  WAIT2                     ;循环执行本语句(停机)，等待看门狗溢出复位
Power_Off_WDT_TIME_LED:
    SETB  WDT_TIME_LED              ;上次亮本次就熄灭看门狗溢出时间指示灯
    CPL   Last_WDT_Time_LED_Status ;将看门狗溢出时间指示灯的上一次状态位取反
WAIT3:
    SJMP  WAIT3                     ;循环执行本语句(停机)，等待看门狗溢出复位
    END
```

4.2 如何用软件实现系统复位

用户应用程序在运行过程当中，有时会有特殊需求，需要实现单片机系统软复位（热启动之一），传统的 8051 单片机由于硬件上未支持此功能，用户必须用软件模拟实现，实现起来较麻烦。现 STC 新推出的增强型 8051 根据客户要求增加了 ISP_CONTR 特殊功能寄存器,实现了此功能。用户只需简单的控制 ISP_CONTR 特殊功能寄存器的其中两位 SWBS / SWRST 就可以系统复位了。

ISP_CONTR：ISP/IAP 控制寄存器，地址在 0E7H 单元

B7	B6	B5	B4	B3	B2	B1	B0	Reset Value
ISPEN	SWBS	SWRST	CMD_FAIL	1	WT2	WT1	WT0	0000,1000

ISPEN： ISP/IAP 功能允许位。0：禁止 ISP/IAP 编程改变 Flash,1:允许编程改变 Flash

SWBS： 软件选择从用户应用程序区启动（0），还是从 ISP 程序区启动（1）。要与 SWRST 直接配合才可以实现

SWRST： 0：不操作； 1： 产生软件系统复位，硬件自动清零。

CMD_FAIL：如果送了 ISP/IAP 命令，并对 ISP_TRIG 送 46h/B9h 触发失败，则为 1，需由软件清零。

```
;从用户应用程序区(AP区)软件复位并切换到用户应用程序区(AP区)开始执行程序
    MOV   ISP_CONTR, #00100000B ;SWBS = 0(选择AP区), SWRST = 1(软复位)
;从系统ISP监控程序区软件复位并切换到用户应用程序区(AP区)开始执行程序
    MOV   ISP_CONTR, #00100000B ;SWBS = 0(选择AP区), SWRST = 1(软复位)

;从用户应用程序区(AP区)软件复位并切换到系统ISP监控程序区开始执行程序
    MOV   ISP_CONTR, #01100000B ;SWBS = 1(选择ISP区), SWRST = 1(软复位)
;从系统ISP监控程序区软件复位并切换到系统ISP监控程序区开始执行程序
    MOV   ISP_CONTR, #01100000B ;SWBS = 1(选择ISP区), SWRST = 1(软复位)
```

本复位是整个系统复位，所有的特殊功能寄存器都会复位到初始值，I/O 口也会初始化。

4.3 热启动复位和冷启动复位

	复位源	现象
热启动复位	内部看门狗复位	会使单片机直接从用户程序区 0000H 处开始执行用户程序
	通过控制 RESET 脚产生的硬复位	会使系统从用户程序区 0000H 处开始直接执行用户程序
	通过对 ISP_CONTR 寄存器送入 20H 产生的软复位	会使系统从用户程序区 0000H 处开始直接执行用户程序
	通过对 ISP_CONTR 寄存器送入 60H 产生的软复位	会使系统从系统 ISP 监控程序区开始执行程序，检测不到合法的 ISP 下载命令流后，会软复位到用户程序区执行用户程序
冷启动复位	系统停电后再上电引起的硬复位	会使系统从系统 ISP 监控程序区开始执行程序，检测不到合法的 ISP 下载命令流后，会软复位到用户程序区执行用户程序

第五章 STC12 系列单片机 EEPROM 的应用

--- 利用 ISP/IAP 技术将内部 Data Flash 当 EEPROM，擦写次数 10 万次以上

--- 5V 单片机工作电压在 3.7V 以上时方可进行 ISP/IAP/EEPROM 操作

--- 3V 单片机工作电压在 2.4V 以上时方可进行 ISP/IAP/EEPROM 操作

5.1 IAP 及 EEPROM 新增特殊功能寄存器介绍

STC12 系列 1T 8051 单片机 ISP/IAP 特殊功能寄存器 ISP/IAP SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ISP_DATA	E2h	ISP/IAP Flash Data Register									1111,1111
ISP_ADDRH	E3h	ISP/IAP Flash Address High									0000,0000
ISP_ADDRL	E4h	ISP/IAP Flash Address Low									0000,0000
ISP_CMD	E5h	ISP/IAP Flash Command Register	-	-	-	-	-	-	MS1	MS0	xxxx,xx00
ISP_TRIG	E6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
ISP_CONTR	E7h	ISP/IAP Control Register	ISPEN	SWBS	SWRST	CMD_FAIL	1	WT2	WT1	WT0	0000,1000

ISP_DATA: ISP/IAP 操作时的数据寄存器。

ISP/IAP 从 Flash 读出的数据放在此处，向 Flash 写的数据也需放在此处

ISP_ADDRH: ISP/IAP 操作时的地址寄存器高八位。

ISP_ADDRL: ISP/IAP 操作时的地址寄存器低八位。

ISP_CMD: ISP/IAP 操作时的命令模式寄存器，须命令触发寄存器触发方可生效。

B7	B6	B5	B4	B3	B2	B1	B0	命令/操作 模式选择
保留						命令		
-	-	-	-	-	-	0	0	Standby 待机模式，无 ISP 操作
-	-	-	-	-	-	0	1	从用户的应用程序区对 "Data Flash/EEPROM区" 进行字节读
-	-	-	-	-	-	1	0	从用户的应用程序区对 "Data Flash/EEPROM区" 进行字节编程
-	-	-	-	-	-	1	1	从用户的应用程序区对 "Data Flash/EEPROM区" 进行扇区擦除

程序在用户应用程序区时，仅可以对数据 Flash 区(EEPROM)进行字节读/字节编程/扇区擦除，STC12C5412AD/12LE5412AD/12C5412/12LE5412/12C5052AD/12LE5052AD/12C5052/12LE5052 除外，这几个型号可在应用程序区修改应用程序区。

ISP_TRIG: ISP/IAP 操作时的命令触发寄存器。

在 ISPEN(ISP_CONTR.7) = 1 时，对 ISP_TRIG 先写入 46h，再写入 B9h，ISP/IAP 命令才会生效。

ISP_CONTR: ISP/IAP 控制寄存器，地址在 0E7H 单元

B7	B6	B5	B4	B3	B2	B1	B0	Reset Value
ISPEN	SWBS	SWRST	CMD_FAIL	1	WT2	WT1	WT0	0000,1000

ISPEN: ISP/IAP 功能允许位。0：禁止 ISP/IAP 编程改变 Flash，1:允许编程改变 Flash

SWBS: 软件选择从用户主程序区启动（0），还是从 ISP 程序区启动（1）。

SWRST: 0：不操作； 1：产生软件系统复位，硬件自动清零。

CMD_FAIL: 如果送了 ISP/IAP 命令，并对 ISP_TRIG 送 46h/B9h 触发失败，则为 1，需由软件清零。

```
;在用户应用程序区(AP区)软件复位并从用户应用程序区(AP区)开始执行程序
    MOV   ISP_CONTR, #00100000B ;SWBS = 0(选择AP区), SWRST = 1(软复位)
;在用户应用程序区(AP区)软件复位并从系统ISP监控程序区开始执行程序
    MOV   ISP_CONTR, #01100000B ;SWBS = 1(选择ISP区), SWRST = 1(软复位)
;在系统ISP监控程序区软件复位并从用户应用程序区(AP区)开始执行程序
    MOV   ISP_CONTR, #00100000B ;SWBS = 0(选择AP区), SWRST = 1(软复位)
;在系统ISP监控程序区软件复位并从系统ISP监控程序区开始执行程序
    MOV   ISP_CONTR, #01100000B ;SWBS = 1(选择ISP区), SWRST = 1(软复位)
```

设置等待时间			CPU 等待时间(多少个 CPU 工作时钟)			
WT2	WT1	WT0	Read/读	Program/编程	Sector Erase 扇区擦除	Recommended System Clock 跟等待参数对应的推荐系统时钟
1	1	1	2个时钟	55个时钟	21012个时钟	〈 1MHz
1	1	0	2个时钟	110个时钟	42024个时钟	〈 2MHz
1	0	1	2个时钟	165个时钟	63036个时钟	〈 3MHz
1	0	0	2个时钟	330个时钟	126072个时钟	〈 6MHz
0	1	1	2个时钟	660个时钟	252144个时钟	〈 12MHz
0	1	0	2个时钟	1100个时钟	420240个时钟	〈 20MHz
0	0	1	2个时钟	1320个时钟	504288个时钟	〈 24MHz
0	0	0	2个时钟	1760个时钟	672384个时钟	〈 30MHz

EEPROM 使用注意事项：

为了保证单片机内部 EEPROM 的正常可靠工作，目前供货的单片机：

5V 单片机在 Vcc<3.7V 时，禁止 ISP/IAP 操作，即禁止对 EEPROM 的正常操作，此时单片机对相应的 ISP/IAP 指令不响应，实际情况是，ISP/IAP 对寄存器的操作是执行了，命令也下达了，但由于此时工作电压低于可靠的门槛电压以下，单片机内部此时禁止执行 ISP/IAP 操作，即对 EEPROM 的擦除 / 编程 / 读命令均无效。

3V 单片机在 Vcc<2.4V 时，禁止 ISP/IAP 操作，即禁止对 EEPROM 的正常操作，此时单片机对相应的 ISP/IAP 指令不响应，实际情况是，ISP/IAP 对寄存器的操作是执行了，命令也下达了，但由于此时工作电压低于可靠的门槛电压以下，单片机内部此时禁止执行 ISP/IAP 操作，即对 EEPROM 的擦除 / 编程 / 读命令均无效。

如果电源上电缓慢，可能会由于程序已经开始运行，而此时电源电压还达不到 EEPROM 的最低可靠工作电压，导致执行相应的 EEPROM 指令无效，所以建议用户在初始化程序中增加 200ms 延时，并判断 LVDF 标志位（PCON 电源管理寄存器的 Bit5 位）。如果该位为“1”，则说明电源电压曾经低于内部检测门槛电压，软件将其清零，加几个空操作延时后再读该位的状态，如果为”0”，说明工作电压在内部低压检测门槛电压以上，则可进行 ISP/IAP/EEPROM 操作。如果为”1”，则将其再清零，一直等到工作电压高于内部低压检测门槛电压以上，才可进行 ISP/IAP/EEPROM 操作。

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000

LVDF：内部电源低电压检测标志位

当单片机工作电压低于检测门槛电压时，该位置‘1”，该位只能通过软件清零，如果允许内部低电压检测中断，则该位为低电压中断请求标志位。

5V 单片机在 Vcc<3.7V 时，LVDF = 1，Vcc 〉3.7V 时，LVDF 的值不变，该位只能通过软件清零；

3V 单片机在 Vcc<2.4V 时，LVDF = 1，Vcc 〉2.4V 时，LVDF 的值不变，该位只能通过软件清零；

5.2 STC12C5410AD 系列单片机 EEPROM 地址

STC12C5410AD 系列和 STC12C2052AD 系列单片机内部可用 Data Flash(EEPROM)的地址(与程序空间是分开的)：如果对应用程序区进行 IAP 写数据 / 擦除扇区的动作，则该语句会被单片机忽略，继续执行下一句。程序在用户应用程序区(AP 区)时,仅可以对 Data Flash(EEPROM)进行 IAP/ISP 操作。

但 STC12C5412AD/12LE5412AD/12C5412/12LE5412/12C5052AD/12LE5052AD/12C5052/12LE5052 在应用程序区可以修改应用程序区(灵活)。

STC12C5410AD系列单片机的内部EEPROM地址表				
第一扇区		第二扇区		每个扇区 512字节,共4个扇区 建议同一次修改的数据放在同一个扇区，不是同一次修改的数据放在不同的扇区，不必用满，当然可全用，用满则为2K字节EEPROM。由于擦除是按扇区擦除，所以每个扇区用的越少越方便，256个字节以内较合理。
起始地址	结束地址	起始地址	结束地址	
2800h	29FFh	2A00h	2BFFh	
第三扇区		第四扇区		
起始地址	结束地址	起始地址	结束地址	
2C00h	2DFFh	2E00h	2FFFh	
适用的型号如下： STC12C5410AD, STC12C5410, STC12LE5410AD, STC12LE5410 STC12C5408AD, STC12C5408, STC12LE5408AD, STC12LE5408 STC12C5406AD, STC12C5406, STC12LE5406AD, STC12LE5406 STC12C5404AD, STC12C5404, STC12LE5404AD, STC12LE5404 STC12C5402AD, STC12C5402, STC12LE5402AD, STC12LE5402 STC12C5401AD, STC12C5401, STC12LE5401AD, STC12LE5401				

STC12C5412,STC12C5412AD,STC12LE5412,STC12LE5412AD 单片机可对自身内部应用程序区进行 IAP/ISP 操作，故所有部分均可当 Data Flash(EEPROM)使用，其地址如下：

第一扇区		第二扇区		第三扇区		第四扇区		每个扇区 512字节 建议同一次修改的数据放在同一个扇区，不是同一次修改的数据放在不同的扇区，不必用满，当然可全用
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
0000h	01FFh	0200h	03FFh	0400h	05FFh	0600h	07FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
0800h	09FFh	0A00h	0BFFh	0C00h	0DFFh	0E00h	0FFFh	
第九扇区		第十扇区		第十一扇区		第十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
1000h	11FFh	1200h	13FFh	1400h	15FFh	1600h	17FFh	
第十三扇区		第十四扇区		第十五扇区		第十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
1800h	19FFh	1A00h	1BFFh	1C00h	1DFFh	1E00h	1FFFh	
第十七扇区		第十八扇区		第十九扇区		第二十扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2000h	21FFh	2200h	23FFh	2400h	25FFh	2600h	27FFh	
第二十一扇区		第二十二扇区		第二十三扇区		第二十四扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2800h	29FFh	2A00h	2BFFh	2C00h	2DFFh	2E00h	2FFFh	

5.3 STC12C2052AD 系列单片机 EEPROM 地址

STC12C2052AD 系列和 STC12C5410AD 系列单片机内部可用 Data Flash(EEPROM)的地址(与程序空间是分开的)：如果对应用程序区进行 IAP 写数据 / 擦除扇区的动作，则该语句会被单片机忽略，继续执行下一句。程序在用户应用程序区(AP 区)时,仅可以对 Data Flash(EEPROM)进行 IAP/ISP 操作。

STC12C2052AD系列单片机的内部EEPROM地址表				
第一扇区		第二扇区		每个扇区 512字节,共2个扇区 建议同一次修改的数据放在同一个扇区，不是同一次修改的数据放在不同的扇区，不必用满，当然可全用，用满则为1K字节EEPROM。由于擦除是按扇区擦除，所以每个扇区用的越少越方便，256个字节以内较合理。
起始地址	结束地址	起始地址	结束地址	
1000h	11FFh	1200h	13FFh	
适用的型号如下： STC12C4052AD，STC12C4052，STC12LE4052AD，STC12LE4052 STC12C2052AD，STC12C2052，STC12LE2052AD，STC12LE2052 STC12C1052AD，STC12C1052，STC12LE1052AD，STC12LE1052 STC12C0552AD，STC12C0552，STC12LE0552AD，STC12LE0552				

STC12C5052,STC12C5052AD,STC12LE5052,STC12LE5052AD 单片机可对自身内部应用程序区进行 IAP/ISP 操作，故所有部分均可当 Data Flash(EEPROM)使用，其地址如下：

第一扇区		第二扇区		第三扇区		第四扇区		每个扇区 512字节 建议同一次修改的数据放在同一个扇区，不是同一次修改的数据放在不同的扇区，不必用满，当然可全用
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
0000h	01FFh	0200h	03FFh	0400h	05FFh	0600h	07FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
0800h	09FFh	0A00h	0BFFh	0C00h	0DFFh	0E00h	0FFFh	
第九扇区		第十扇区						
起始地址	结束地址	起始地址	结束地址					
1000h	11FFh	1200h	13FFh					

5.4 IAP 及 EEPROM 汇编简介

```
;用 DATA 还是 EQU 声明新增特殊功能寄存器地址要看你用的汇编器 / 编译器
    ISP_DATA        DATA    0E2h; 或    ISP_DATA        EQU     0E2h
    ISP_ADDRH       DATA    0E3h; 或    ISP_ADDRH       EQU     0E3h
    ISP_ADDRL       DATA    0E4h; 或    ISP_ADDRL       EQU     0E4h
    ISP_CMD         DATA    0E5h; 或    ISP_CMD         EQU     0E5h
    ISP_TRIG        DATA    0E6h; 或    ISP_TRIG        EQU     0E6h
    ISP_CONTR       DATA    0E7h; 或    ISP_CONTR       EQU     0E7h
;定义 ISP/IAP 命令及等待时间
    ISP_IAP_BYTE_READ     EQU   1     ;字节读
    ISP_IAP_BYTE_PROGRAM  EQU   2     ;字节编程,前提是该字节是空，0FFh
    ISP_IAP_SECTOR_ERASE  EQU   3     ;扇区擦除,要某字节为空，要擦一扇区
    WAIT_TIME       EQU   0           ;设置等待时间,30MHz 以下 0,24M 以下 1,
        ;20MHz 以下 2,12M 以下 3,6M 以下 4,3M 以下 5,2M 以下 6,1M 以下 7,
;字节读
    MOV   ISP_ADDRH, #BYTE_ADDR_HIGH      ;送地址高字节
    MOV   ISP_ADDRL, #BYTE_ADDR_LOW       ;送地址低字节
                                          地址需要改变时才需重新送地址
    MOV   ISP_CONTR, #WAIT_TIME ;设置等待时间
    ORL   ISP_CONTR, #10000000B ;允许 ISP/IAP 操作
                                          此两句可以合成一句,并且只送一次就够了
    MOV   ISP_CMD,        #ISP_IAP_BYTE_READ;送字节读命令,命令不需改变时,不需重新送命令
    MOV   ISP_TRIG,       #46h        ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV   ISP_TRIG,       #0B9h ;送完 B9h 后，ISP/IAP 命令立即被触发起动
;CPU 等待 IAP 动作完成后，才会继续执行程序。
    NOP                           ;数据读出到 ISP_DATA 寄存器后，CPU 继续执行程序
    MOV   A,        ISP_DATA      ;将读出的数据送往 Acc
;以下语句可不用,只是出于安全考虑而已
    MOV   ISP_CONTR, #00000000B   ;禁止 ISP/IAP 操作
    MOV   ISP_CMD,   #00000000B   ;去除 ISP/IAP 命令
    ;MOV  ISP_TRIG,  #00000000B   ;防止 ISP/IAP 命令误触发
    ;MOV  ISP_ADDRH, #0           ;送地址高字节单元为 00,指向非 EEPROM 区
    ;MOV  ISP_ADDRL, #0           ;送地址低字节单元为00,防止误操作
```

```
;字节编程，该字节为 FFh/ 空时，可对其编程，否则不行,要先执行扇区擦除
    MOV   ISP_DATA,  #ONE_DATA    ;送字节编程数据到ISP_DATA,只有数据改变时才需重新送
    MOV   ISP_ADDRH, #BYTE_ADDR_HIGH ;送地址高字节
    MOV   ISP_ADDRL, #BYTE_ADDR_LOW  ;送地址低字节      地址需要改变时才需重新送地址
    MOV   ISP_CONTR, #WAIT_TIME ;设置等待时间
    ORL   ISP_CONTR, #10000000B ;允许 ISP/IAP 操作     此两句可合成一句,并且只送一次就够了
    MOV   ISP_CMD,   #ISP_IAP_BYTE_PROGRAM       ;送字节编程命令
    MOV   ISP_TRIG,  #46h          ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV   ISP_TRIG,  #0B9h         ;送完 B9h 后，ISP/IAP 命令立即被触发起动
;CPU 等待 IAP 动作完成后，才会继续执行程序.
    NOP                            ;字节编程成功后，CPU 继续执行程序
;以下语句可不用,只是出于安全考虑而已
    MOV   ISP_CONTR, #00000000B ;禁止 ISP/IAP 操作
    MOV   ISP_CMD,   #00000000B ;去除 ISP/IAP 命令
    ;MOV  ISP_TRIG,  #00000000B ;防止 ISP/IAP 命令误触发
    ;MOV  ISP_ADDRH, #0          ;送地址高字节单元为 00,指向非 EEPROM 区,防止误操作
    ;MOV  ISP_ADDRL, #0          ;送地址低字节单元为 00,指向非 EEPROM 区,防止误操作

;扇区擦除，没有字节擦除，只有扇区擦除，512 字节 / 扇区,每个扇区用得越少越方便
;如果要对某个扇区进行擦除，而其中有些字节的内容需要保留，则需将其先读到单片机
;内部的 RAM 中保存，再将该扇区擦除，然后将须保留的数据写回该扇区，所以每个扇区
;中用的字节数越少越好，操作起来越灵活越快.
;扇区中任意一个字节的地址都是该扇区的地址,无需求出首地址.

    MOV   ISP_ADDRH, #SECTOR_FIRST_BYTE_ADDR_HIGH ;送扇区起始地址高字节   地址需要改变时
    MOV   ISP_ADDRL, #SECTOR_FIRST_BYTE_ADDR_LOW  ;送扇区起始地址低字节   才需重新送地址
    MOV   ISP_CONTR, #WAIT_TIME ;设置等待时间
    ORL   ISP_CONTR, #10000000B ;允许 ISP/IAP      此两句可以合成一句,并且只送一次就够了
    MOV   ISP_CMD,   #ISP_IAP_SECTOR_ERASE;送扇区擦除命令,命令不需改变时,不需重新送命令
    MOV   ISP_TRIG,        #46h        ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV   ISP_TRIG,        #0B9h       ;送完 B9h 后，ISP/IAP 命令立即被触发起动
;CPU 等待 IAP 动作完成后，才会继续执行程序.
    NOP                                ;扇区擦除成功后，CPU 继续执行程序
;以下语句可不用,只是出于安全考虑而已
    MOV   ISP_CONTR, #00000000B        ;禁止 ISP/IAP 操作
    MOV   ISP_CMD,   #00000000B        ;去除 ISP/IAP 命令
    ;MOV  ISP_TRIG,  #00000000B        ;防止 ISP/IAP 命令误触发
    ;MOV  ISP_ADDRH, #0                ;送地址高字节单元为 00,指向非 EEPROM 区
    ;MOV  ISP_ADDRL, #0                ;送地址低字节单元为00,防止误操作
```

小常识： (STC 单片机的 Data Flash 当 EEPROM 功能使用)

3 个基本命令 - - - - 字节读，字节编程，扇区擦除

字节编程：只能将“1”改为“0”，对“0”用字节编程是无用的。如果该字节是“1111,1111B”，则可将其中的“1”编程为“0”，如果该字节中有位为“0”，要将其改为“1”，则须先将整个扇区擦除，因为只有“扇区擦除”才可以将“0”变为“1”。

扇区擦除：只有“扇区擦除”才可能将“0”擦除为“1”。

大建议：

1.同一次修改的数据放在同一扇区中，不是同一次修改的数据放在另外的扇区,就不须读出保护。

2.如果一个扇区只用一个字节，那就是真正的 EEPROM,STC 单片机的 Data Flash 比外部 EEPROM 要快很多，读一个字节 / 编程一个字节大概是 0.2uS/60uS。

3.如果在一个扇区中存放了大量的数据，某次只需要修改其中的一个字节或一部分字节时，则另外的不需要修改的数据须先读出放在 STC 单片机的 RAM 中，然后擦除整个扇区，再将需要保留的数据和需修改的数据一并写回该扇区中。这时每个扇区使用的字节数是使用的越少越方便(不需读出一大堆需保留数据)。

5.5 一个完整的 EEPROM 测试程序，用宏晶的下载板可以直接测试

```
;STC12C5410AD 系列和 STC12C2052AD 系列单片机 EEPROM/IAP 功能测试程序演示
;/* ---  STC International Limited ---------------- */
;/* ---  宏晶科技 姚永平  设计 2006/1/6   V1.0 ----- */
;/* ---  演示 STC12C5410AD 系列 MCU EEPROM/IAP 功能 - */
;/* ---  演示 STC12C2052AD 系列 MCU EEPROM/IAP 功能 - */
;/* ---  Mobile: 13922805190 ---------------------- */
;/* ---  Fax: 0755-82944243 ----------------------- */
;/* ---  Tel: 0755-82948409 ----------------------- */
;/* ---  Web: www.mcu-memory.com ------------------ */
;本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过,EEPROM 的数据在 P1 口上显示
;如果要在程序中使用或在文章中引用该程序,请在程序中或文章中注明使用了宏晶科技的资料及程序
;-----------------------------------------------
;声明与 IAP/ISP/EEPROM 有关的特殊功能寄存器的地址
ISP_DATA        EQU   0E2H
ISP_ADDRH       EQU   0E3H
ISP_ADDRL       EQU   0E4H
ISP_CMD         EQU   0E5H
ISP_TRIG        EQU   0E6H
ISP_CONTR       EQU   0E7H
;定义 ISP/IAP 命令
ISP_IAP_BYTE_READ       EQU   1H  ;字节读
ISP_IAP_BYTE_PROGRAM    EQU   2H  ;字节编程,可以将 1 写成 0，要将 1 变成 0，必须执行字节编程
ISP_IAP_SECTOR_ERASE    EQU   3H  ;扇区擦除,可以将 0 擦成 1，要将 0 变成 1，必须擦除整个扇区
;定义 Flash 操作等待时间及允许 IAP/ISP/EEPROM 操作的常数
;ENABLE_ISP      EQU   80H      ;系统工作时钟<30MHz 时，对 ISP_CONTR 寄存器设置此值
;ENABLE_ISP      EQU   81H      ;系统工作时钟<24MHz 时，对 ISP_CONTR 寄存器设置此值
 ENABLE_ISP      EQU   82H      ;系统工作时钟<20MHz 时，对 ISP_CONTR 寄存器设置此值
;ENABLE_ISP      EQU   83H      ;系统工作时钟<12MHz 时，对 ISP_CONTR 寄存器设置此值
;ENABLE_ISP      EQU   84H      ;系统工作时钟<6MHz 时，对 ISP_CONTR 寄存器设置此值
;ENABLE_ISP      EQU   85H      ;系统工作时钟<3MHz 时，对 ISP_CONTR 寄存器设置此值
;ENABLE_ISP      EQU   86H      ;系统工作时钟<2MHz 时，对 ISP_CONTR 寄存器设置此值
;ENABLE_ISP      EQU   87H      ;系统工作时钟<1MHz 时，对 ISP_CONTR 寄存器设置此值
DEBUG_DATA      EQU   5AH ;是本测试程序选定的 EEPROM 单元的数值如正确应等于的数值
;------------------------
;选择 MCU 型号
;DATA_FLASH_START_ADDRESS EQU 1000H ;STC12C2052AD 系列单片机的 EEPROM 测试起始地址
DATA_FLASH_START_ADDRESS EQU 2800H  ;STC12C5410AD 系列单片机的 EEPROM 测试起始地址
;-----------------------------------------------
    ORG   0000H
    LJMP  MAIN
;-----------------------------------------------
    ORG   0100H
MAIN:
    MOV     P1,#0F0H        ;演示程序开始工作,让 P1.0/P1.1/P1.2/P1.3 控制的灯亮
    LCALL   Delay           ;延时
    MOV     P1,#0FH         ;演示程序开始工作,让 P1.7/P1.6/P1.5/P1.4 控制的灯亮
```

```
    LCALL   Delay           ;延时
    MOV     SP,  #7FH       ;堆栈指针指向 7FH 单元
;************************************************
;将 EEPROM 测试起始地址单元的内容读出
MAIN1:
    MOV     DPTR, #DATA_FLASH_START_ADDRESS   ;将 EEPROM 测试起始地址送 DPTR 数据指针
    LCALL   Byte_Read
    MOV     40H, A                  ;将 EEPROM 的值送 40H 单元保存
    CJNE    A, #DEBUG_DATA, DATA_NOT_EQU_DEBUG_DATA    ;如果数据比较不正确,就跳转

DATA_IS_DEBUG_DATA:
;数据是对的,亮 P1.7 控制的灯,然后在 P1 口上将 EEPROM 的数据显示出来
    MOV     P1,  #01111111B   ;如 (DATA_FLASH_START_ADDRESS)的值等于 #DEBUG_DATA, 亮 P1.7
    LCALL   Delay         ;延时
    MOV     A, 40H    ;将保存在 40H 单元中 EEPROM 的值从 40H 单元送累加器 A
    CPL     A         ;取反的目的是相应的灯亮代表 1,不亮代表 0
    MOV     P1,A      ;数据是对的，送 P1 显示
WAIT1:
    SJMP    WAIT1     ;数据是对的，送 P1 显示后，CPU 在此无限循环执行此句

DATA_NOT_EQU_DEBUG_DATA:
;EEPROM 里的数据是错的,亮 P1.3 控制的灯,然后在 P1 口上将错误的数据显示出来,
;再将该 EEPROM 所在的扇区整个擦除,将正确的数据写入后,亮 P1.5 控制的灯
    MOV     P1,  #11110111B ;如 (DATA_FLASH_START_ADDRESS)的值不等于 #DEBUG_DATA, 亮 P1.3
    LCALL   Delay       ;延时
    MOV     A,    40H   ;将保存在 40H 单元中 EEPROM 的值从 40H 单元送累加器 A
    CPL     A           ;取反的目的是相应的灯亮代表 1,不亮代表 0
    MOV     P1,   A     ;数据不对，送 P1 显示
    LCALL   Delay       ;延时

    MOV     DPTR,#DATA_FLASH_START_ADDRESS   ;将 EEPROM 测试起始地址送 DPTR 数据指针
    ACALL   Sector_Erase   ;擦除整个扇区
    MOV     DPTR, #DATA_FLASH_START_ADDRESS   ;将 EEPROM 测试起始地址送 DPTR 数据指针
    MOV     A,     #DEBUG_DATA        ;写入 EEPROM 的数据为 #DEBUG_DATA
    ACALL   Byte_Program               ;字节编程
    MOV     P1,    #11011111B         ;将先前亮的 P1.3 灯关闭 ,再亮 P1.5 灯,代表数据已被修改
WAIT2:
    SJMP    WAIT2         ;字节编程后,CPU 在此无限循环执行此句
;************************************************
```

```
;-----------------------------------------------
;读一字节，调用前需打开 IAP 功能，入口:DPTR = 字节地址，返回:A = 读出字节
Byte_Read:
    MOV     ISP_CONTR,  #ENABLE_ISP      ;打开 IAP 功能，设置 Flash 操作等待时间
    MOV     ISP_CMD,    #ISP_IAP_BYTE_READ   ;设置为 IAP/ISP/EEPROM 字节读模式命令
    MOV     ISP_ADDRH,  DPH              ;设置目标单元地址的高 8 位地址
    MOV     ISP_ADDRL,  DPL              ;设置目标单元地址的低 8 位地址
    ;CLR     EA
    MOV     ISP_TRIG,   #46H        ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV     ISP_TRIG,   #0B9H       ;送完 B9h 后，ISP/IAP 命令立即被触发起动
    NOP
    MOV     A,    ISP_DATA          ;读出的数据在 ISP_DATA 单元中,送入累加器 A
    ;SETB    EA
    ACALL IAP_Disable ;关闭 IAP 功能，清相关的特殊功能寄存器,使 CPU 处于安全状态,
                      ;一次连续的 IAP 操作完成之后建议关闭 IAP 功能,不需要每次都关
    RET
;-----------------------------------------------
;字节编程，调用前需打开 IAP 功能，入口:DPTR = 字节地址，A= 须编程字节的数据
Byte_Program:
    MOV   ISP_CONTR, #ENABLE_ISP     ;打开 IAP 功能，设置 Flash 操作等待时间
    MOV   ISP_CMD, #ISP_IAP_BYTE_PROGRAM    ;设置为 IAP/ISP/EEPROM 字节编程模式命令
    MOV   ISP_ADDRH, DPH             ;设置目标单元地址的高 8 位地址
    MOV   ISP_ADDRL, DPL             ;设置目标单元地址的低 8 位地址
    MOV   ISP_DATA, A                ;要编程的数据先送进 ISP_DATA 寄存器
    ;CLR   EA
    MOV   ISP_TRIG, #46H             ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV   ISP_TRIG, #0B9H            ;送完 B9h 后，ISP/IAP 命令立即被触发起动
    NOP
    ;SETB  EA
     ACALL IAP_Disable ;关闭 IAP 功能，清相关的特殊功能寄存器,使 CPU 处于安全状态,
                       ;一次连续的 IAP 操作完成之后建议关闭 IAP 功能,不需要每次都关
    RET
;-----------------------------------------------
;擦除扇区，入口:DPTR = 扇区地址
Sector_Erase:
    MOV   ISP_CONTR, #ENABLE_ISP     ;打开 IAP 功能，设置 Flash 操作等待时间
    MOV   ISP_CMD, #03H              ;设置为 IAP/ISP/EEPROM 扇区擦除模式命令
    MOV   ISP_ADDRH, DPH             ;设置目标单元地址的高 8 位地址
    MOV   ISP_ADDRL, DPL             ;设置目标单元地址的低 8 位地址
    ;CLR   EA
    MOV   ISP_TRIG, #46H             ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器,每次都需如此
    MOV   ISP_TRIG, #0B9H            ;送完 B9h 后，ISP/IAP 命令立即被触发起动
    NOP
    ;SETB  EA
     ACALL IAP_Disable ;关闭 IAP 功能，清相关的特殊功能寄存器,使 CPU 处于安全状态,
                       ;一次连续的 IAP 操作完成之后建议关闭 IAP 功能,不需要每次都关
    RET
```

```
;-------------------------------------------------
IAP_Disable:
;关闭 IAP 功能，清相关的特殊功能寄存器,使CPU处于安全状态,
;一次连续的IAP操作完成之后建议关闭IAP功能,不需要每次都关
    MOV   ISP_CONTR, #0              ;关闭 IAP 功能
    MOV   ISP_CMD,  #0               ;清命令寄存器,使命令寄存器无命令,此句可不用
    MOV   ISP_TRIG, #0               ;清命令触发寄存器,使命令触发寄存器无触发,此句可不用
    RET
;-------------------------------------------------
Delay:
    CLR   A
    MOV   R0, A
    MOV   R1, A
    MOV   R2, #20H
Delay_Loop:
    DJNZ  R0, Delay_Loop
    DJNZ  R1, Delay_Loop
    DJNZ  R2, Delay_Loop
    RET

;-------------------------------------------------

    END
;*************************************************
```

第六章 STC12 系列单片机定时器应用

6.1 定时器 0/1 的介绍

STC12C5410AD 系列和 STC12C2052AD 系列有定时器 0 和定时器 1 两个 16 位定时器，与传统 8051 的定时器完全兼容，也可以设置为 1T 模式，其中在定时器 1 做波特率发生器时，定时器 0 可以当两个 8 位定时器用。

定时器 0 和 1

定时和计数功能由特殊功能寄存器 TMOD 的控制位 C/$\overline{T}$ 进行选择，TMOD 寄存器的各位信息如下表所列。可以看出，2 个定时 / 计数器有 4 种操作模式，通过 TMOD 的 M1 和 M0 选择。2 个定时 / 计数器的模式 0、1 和 2 都相同，模式 3 不同，各模式下的功能如下所述。

寄存器 TMOD 各位的功能描述

TMOD 地址：89H 复位值：00H

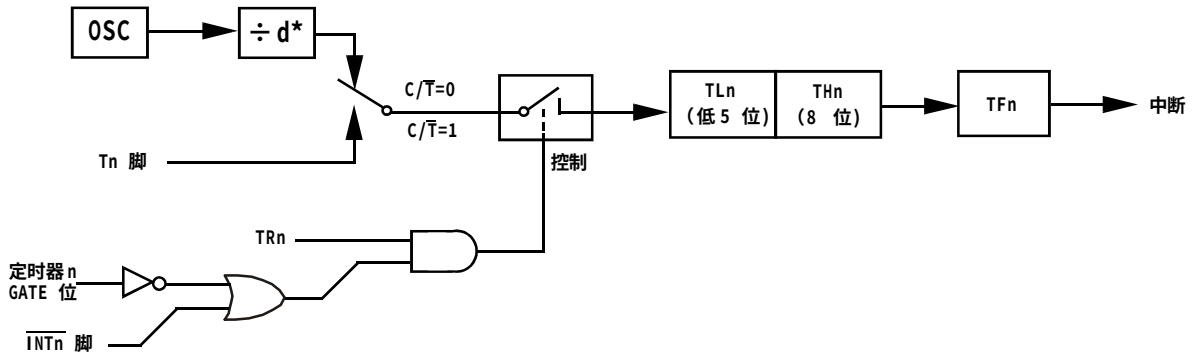
不可位寻址

7	6	5	4	3	2	1	0
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
定时器 1				定时器 0			

位	符号	功能
TMOD.7/	GATE	TMOD.7 控制定时器 1,置 1 时只有在 $\overline{INT1}$ 脚为高及 TR1 控制位置 1 时才可打开定时器 / 计数器 1。
TMOD.3/	GATE	TMOD.3 控制定时器 0,置 1 时只有在 $\overline{INT0}$ 脚为高及 TR0 控制位置 1 时才可打开定时器 / 计数器 0。
TMOD.6/	C/$\overline{T}$	TMOD.6 控制定时器 1 用作定时器或计数器，清零则用作定时器（从内部系统时钟输入），置 1 用作计数器（从 T1/P3.5 脚输入）
TMOD.2/	C/$\overline{T}$	TMOD.2 控制定时器 0 用作定时器或计数器，清零则用作定时器（从内部系统时钟输入），置 1 用作计数器（从 T0/P3.4 脚输入）
TMOD.5/TMOD.4	M1、M0	定时器定时器 / 计数器 1 模式选择
	0 0	13 位定时器 / 计数器，兼容 8048 定时器模式，TL1 只用低 5 位参与分频，TH1 整个 8 位全用。
	0 1	16 位定时器 / 计数器，TL1、TH1 全用
	1 0	8 位自动重装载定时器，当溢出时将 TH1 存放的值自动重装入 TL1。
	1 1	定时器 / 计数器 1 此时无效（停止计数）。
TMOD.1/TMOD.0	M1、M0	定时器 / 计数器 0 模式选择
	0 0	13 位定时器 / 计数器，兼容 8048 定时器模式，TL0 只用低 5 位参与分频，TH0 整个 8 位全用。
	0 1	16 位定时器 / 计数器，TL0、TH0 全用
	1 0	8 位自动重装载定时器，当溢出时将 TH0 存放的值自动重装入 TL0。
	1 1	定时器 0 此时作为双 8 位定时器 / 计数器。TL0 作为一个 8 位定时器 / 计数器，通过标准定时器 0 的控制位控制。TH0 仅作为一个 8 位定时器，由定时器 1 的控制位控制。

1. 模式 0

将定时器设置成模式 0 时类似 8048 定时器，即 8 位计数器带 32 分频的预分频器。下图所示为模式 0 工作方式。此模式下，定时器配置为 13 位的计数器，由 TLn 的低 5 位和 THn 的 8 位所构成。TLn 低 5 位溢出向 THn 进位，THn 计数溢出置位 TCON 中的溢出标志位 TFn（n=0，1）。GATE=0 时，如 TRn=1，则定时器计数。GATE=1 时，允许由外部输入 $\overline{INT1}$ 控制定时器 1，$\overline{INT0}$ 控制定时器 0，这样可实现脉宽测量。TRn 为 TCON 寄存器内的控制位，TCON 寄存器各位的具体功能描述见 TCON 寄存器各位的具体功能描述表。



* 在 T0x12 = 0 模式下，d=12(12 时钟模式)； 在 T0x12 = 1 模式下，d=1(1T)。

图 定时器 / 计数器 0 和定时器 / 计数器 1 的模式 0 ： 13 位定时 / 计数器

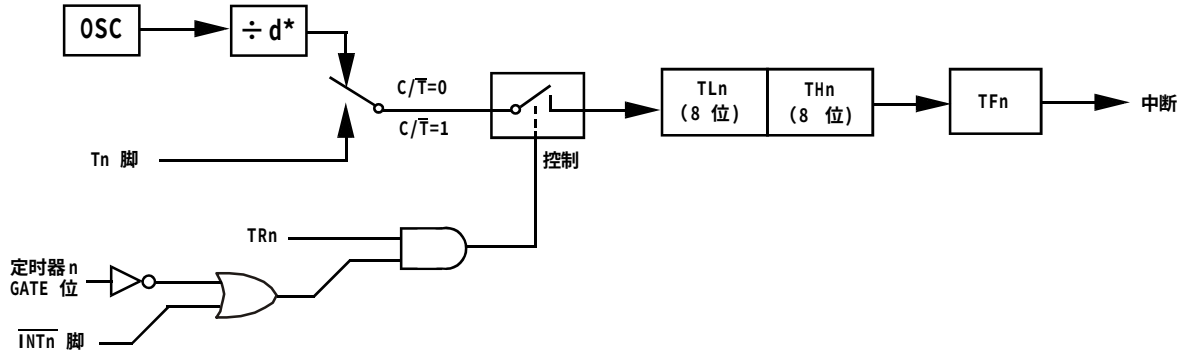
表 寄存器 TCON 各位的功能描述

TCON 地址：88H									
可位寻址		7	6	5	4	3	2	1	0
复位值：00H		TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
位	符 号	功 能							
TCON.7	TF1	定时器 / 计数器 1 溢出标志位。当 T1 被允许计数后，T1 从初值开始加 1 计数，最高位产生溢出时，置“1”TF1，并向 CPU 请求中断，当 CPU 响应时，由硬件清“0”TF1，TF1 也可以由程序查询或清“0”。							
TCON.6	TR1	定时器 T1 的运行控制位。该位由软件置位和清零。当 GATE（TMOD.7）=0，TR1=1 时就允许 T1 开始计数，TR1=0 时禁止 T1 计数。当 GATE（TMOD.7）=1，TR1=1 且 INT1 输入高电平时，才允许 T1 计数。							
TCON.5	TF0	定时器 / 计数器 0 溢出标志位。当 T0 被允许计数后，T0 从初值开始加 1 计数，最高位产生溢出时，置“1”TF0，并向 CPU 请求中断，当 CPU 响应时，由硬件清“0”TF0，TF0 也可以由程序查询或清“0”。							
TCON.4	TR0	定时器 T0 的运行控制位。该位由软件置位和清零。当 GATE（TMOD.3）=0，TR0=1 时就允许 T0 开始计数，TR1=0 时禁止 T0 计数。当 GATE（TMOD.3）=1，TR0=1 且 $\overline{INT0}$ 输入高电平时，才允许 T0 计数。							
TCON.3	IE1	外部中断 1 中断请求标志位。当主机响应中断转向该中断服务程序执行时，由内部硬件自动将 IE1 位清 0。							
TCON.2	IT1	外部中断 1 触发方式控制位。IT1=0 时，外部中断 1 为低电平触发方式，当 $\overline{INT1}$（P3.3）输入低电平时，置位 IE1。采用低电平触发方式时，外部中断源（输入到 INT1）必须保持低电平有效，直到该中断被 CPU 响应，同时在该中断服务程序执行完之前，外部中断源必须被清除（P3.3 要变高），否则将产生另一次中断。当 IT1=1 时，则外部中断 1（$\overline{INT1}$）端口由“1”→“0”下降沿跳变，激活中断请求标志位 IE1，向主机请求中断处理。							
TCON.1	IE0	外部中断 0 中断请求标志位。当主机响应中断转向该中断服务程序执行时，由内部硬件自动将 IE0 位清 0。							
TCON.0	IT0	外部中断 0 触发方式控制位。IT0=0 时，外部中断 0 为低电平触发方式，当 $\overline{INT0}$（P3.2）输入低电平时，置位 IE0。采用低电平触发方式时，外部中断源（输入到 $\overline{INT0}$）必须保持低电平有效，直到该中断被 CPU 响应，同时在该中断服务程序执行完之前，外部中断源必须被清除（P3.2 要变高），否则将产生另一次中断。当 IT0=1 时，则外部中断 0（$\overline{INT0}$）端口由“1”→“0”下降沿跳变，激活中断请求标志位 IE1，向主机请求中断处理。							

该 13 位寄存器包含 THn 全部 8 个位及 TLn 的低 5 位。TLn 的高 3 位不定，可将其忽略。置位运行标志（TRn）不能清零此寄存器。模式 0 的操作对于定时器 0 及定时器 1 都是相同的。2 个不同的 GATE 位（TMOD.7 和 TMOD.3）分别分配给定时器 1 及定时器 0。

2 . 模式 1

模式 1 除了使用了 THn 及 TLn 全部 16 位外，其他与模式 0 完全相同。

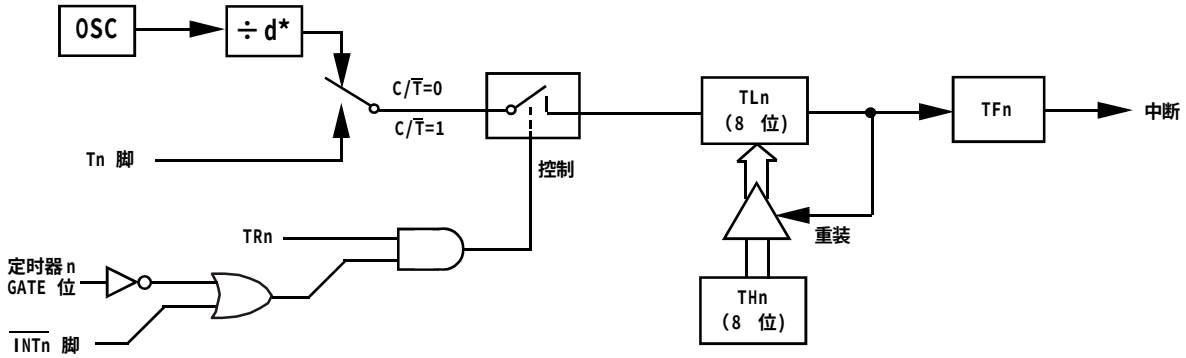


* 在 T0x12 = 0 模式下，d=12(12 时钟模式)； 在 T0x12 = 1 模式下，d=1(1T)。

图 定时器 / 计数器 0 和定时器 / 计数器 1 的模式 1 ： 16 位定时 / 计数器

3 . 模式 2

此模式下定时器 / 计数器 0 和 1 作为可自动重装载的 8 位计数器（TLn），如下图所示。TLn 的溢出不仅置位 TFn，而且将 THn 内容重新装入 TLn，THn 内容由软件预置，重装时 THn 内容不变。模式 2 的操作对于定时器 0 及定时器 1 是相同的。



* 在 T0x12 = 0 模式下，d=12(12 时钟模式)； 在 T0x12 = 1 模式下，d=1(1T)。

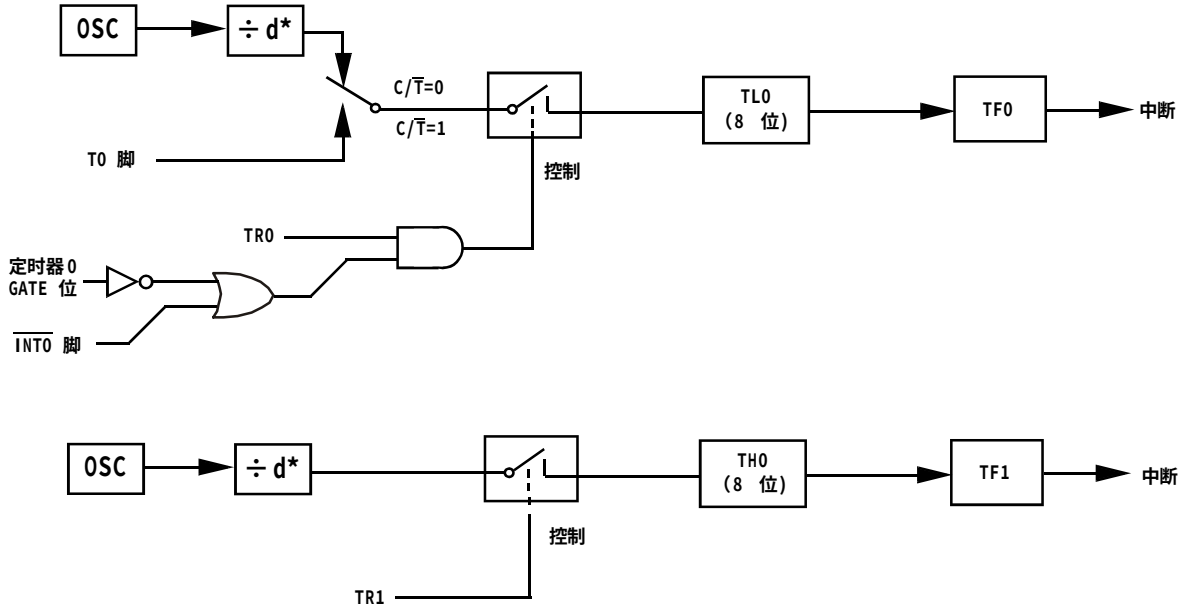
图 定时器 / 计数器 0 和 1 的模式 2 ： 8 位自动重装载

4 . 模式 3

对定时器 1，在模式 3 时，定时器 1 停止计数，效果与将 TR1 设置为 0 相同。

对定时器 0，此模式下定时器 0 的 TL0 及 TH0 作为 2 个独立的 8 位计数器。下图为模式 3 时的定时器 0 逻辑图。TL0 占用定时器 0 的控制位：C/$\overline{T}$、GATE、TR0、$\overline{INT0}$ 及 TF0。TH0 限定为定时器功能（计数器周期），占用定时器 1 的 TR1 及 TF1。此时，TH0 控制定时器 1 中断。

模式 3 是为了增加一个附加的 8 位定时器 / 计数器而提供的，使单片机具有三个定时器 / 计数器。模式 3 只适用于定时器 / 计数器 0，定时器 T1 处于模式 3 时相当于 TR1=0，停止计数（此时 T1 可用来作串行口波特率发生器），而 T0 可作为两个定时器用。



* 在 T0x12 = 0 模式下，d=12(12 时钟模式)； 在 T0x12 = 1 模式下，d=1(1T)。

图 定时 / 计数器 0 的模式 3 ： 两个 8 位计数器

5 . 也可将定时器 0 和定时器 1 设置为 1T 模式

STC12C5410AD 和 STC12C2052AD 系列是 1T 的 8051 单片机，为兼容传统 8051，定时器 0 和定时器 1 复位后是传统 8051 的速度，即 12 分频，这是为了兼容传统 8051。但也可不进行 12 分频，通过设置新增加的特殊功能寄存器 AUXR,将 T0,T1 设置为 1T。普通 111 条机器指令是固定的，快 3 到 24 倍，无法改变。

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx

T0x12： 0，定时器 0 是传统 8051 速度，12 分频；1， 定时器 0 的速度是传统 8051 的 12 倍，不分频
T1x12： 0，定时器 1 是传统 8051 速度，12 分频；1， 定时器 1 的速度是传统 8051 的 12 倍，不分频
如果 UART 串口用定时器 1 做波特率发生器，T1x12 位就可以控制 UART 串口是 12T 还是 1T 了。

UART 串口的模式 0：
STC12C5410AD 和 STC12C2052AD 系列是 1T 的 8051 单片机，为了兼容传统 8051，UART 串口复位后是兼容传统 8051 的。
UART_M0x6： 0，UART 串口的模式 0 是传统 12T 的 8051 速度，12 分频；
1，UART 串口的模式 0 的速度是传统 12T 的 8051 的 6 倍，2 分频
如果用定时器 T1 做波特率发生器时,UART 串口的速度由 T1 的溢出率决定

6.2 定时器0/1应用举例

【例1】 定时/计数器编程，定时/计数器的应用编程主要需考虑：根据应用要求，通过程序初始化，正确设置控制字，正确计算和计算计数初值，编写中断服务程序，适时设置控制位等。通常情况下，设置顺序大致如下：

1)工作方式控制字（TMOD、T2CON）的设置；

2)计数初值的计算并装入THx、TLx、RCAP2H、RCAP2L；

3)中断允许位ETx、EA的设置，使主机开放中断；

4) 启/停位TRx的设置等。

现以定时/计数器0或1为例作一简要介绍。

8051系列单片机的定时器/计数器0或1是以不断加1进行计数的，即属加1计数器，因此，就不能直接将实际的计数值作为计数初值送入计数寄存器THx、TLx中去，而必须将实际计数值以2^8、2^{13}、2^{16}为模求补，以其补码作为计数初值设置THx和TLx。

设：实际计数值为X，计数器长度为n（n=8、13、16），则应装入计数器THx、TLx中的计数初值为2^n-x，式中2^n为取模值。例如，工作方式0的计数长度为13位，则n=13，以2^{13}为模，工作方式1的计数长度为16，则n=16，以2^{16}为模等等。所以，计数初值为（x）= 2^n-x。

对于定时模式，是对机器周期计数，而机器周期与选定的主频密切相关。因此，需根据应用系统所选定的主频计算出机器周期值。现以主频6MHz为例，则机器周期为：

$$\text{一个机器周期} = \frac{12}{\text{主振频率}} = \frac{12}{6\times10^6}\ \mu s = 2\ \mu s$$

实际定时时间Tc = x·Tp

式中Tp为机器周期，Tc为所需定时时间，x为所需计数次数。Tp和Tp一般为已知值，在求出Tp后即可求得所需计数值x，再将x求补码，即求得定时计数初值。即

（x）补= 2^n - x

例如，设定时时间Tc = 5ms，机器周期TP = 2 μs，可求得定时计数次数

$$x = \frac{5ms}{2\ \mu s} = 2500\text{次}$$

设选用工作方式1，则n=16，则应设置的定时时间计数初值为：（x）补= 2^{16} - x=65536-2500=63036，还需将它分解成两个8位十六进制数，分别求得低8位为3CH装入TLx，高8位为F6H装入THx中。

工作方式0、1、2的最大计数次数分别为8192、65536和256。

对外部事件计数模式，只需根据实际计数次数求补后变换成两个十六进制码即可。

【例2】 定时/计数器应用编程，设某应用系统，选择定时/计数器1定时模式，定时时间Tc = 10ms，主频频率为12MHz，每10ms向主机请求处理。选定工作方式1。计算得计数初值：低8位初值为F0H，高8位初值为D8H。

（1）初始化程序

所谓初始化，一般在主程序中根据应用要求对定时/计数器进行功能选择及参数设定等预置程序，本例初始化程序如下：

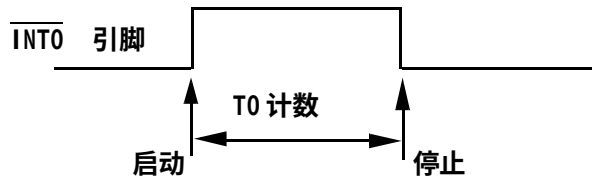
```
START：
        ⋮                       ；主程序段
        MOV  SP，#60H           ；设置堆栈区域
        MOV  TMOD，#10H         ；选择T1、定时模式，工作方式1
        MOV  TH1，#0D8H         ；设置高字节计数初值
        MOV  TL1，#0F0H         ；设置低字节计数初值
        SETB EA                 }
        SETB ET1                }  开中断
        ⋮                       ；其他初始化程序
        SETB TR1                ；启动T1开始计时
        ⋮                       ；继续主程序
```

（2）中断服务程序

```
INTT1： PUSH A                  }
        PUSH DPL                }  现场保护
        PUSH DPH                }
        ⋮
        MOV  TL1,#0F0H          }
        MOV  TH1,#0D8H          }  重新置初值
        ⋮                       ；中断处理主体程序
        POP  DPH                }
        POP  DPL                }  现场恢复
        POP  A                  }
        RETI                    ；返回
```

这里展示了中断服务子程序的基本格式。8052系列单片机的中断属于矢量中断，每一个矢量中断源只留有8个字节单元，一般是不够用的，常需用转移指令转到真正的中断服务子程序区去执行。

【例3】 对外部正脉冲测宽。选择定时/计数器2进行脉宽测试较方便，但也可选用定时/计数器0或定时/计数器1进行测宽操作。本例选用定时/计数器0（T0）以定时模式，工作方式1对$\overline{INT0}$引脚上的正脉冲进行脉宽测试。



设置GATE为1，机器周期TP为1 μs。本例程序段编制如下：

```
INTT0： MOV  TMOD，#09H         ；设T0为定时方式1，GATE为1
```

```
            MOV   TL0，#00H          ;
                                     } TH0,TL0 清 0
            MOV   TH0，#00H          ;
            CLR   EX0                ；关 INT0 中断
LOP1：      JB    P3.2，LOP1         ；等待 INT0 引低电平
LOP2：      JNB   P3.2，LOP2         ；等待 INT0 引脚高电平
            SETB  TR0                ；启动 T0 开始计数
LOP3：      JB    P3.2，LOP3         ；等待 INT0 低电平
            CLR   TR0                ；停止 T0 计数
            MOV   A，TL0             ；低字节计数值送 A
            MOV   B，TH0             ；高字节计数值送 B
             ⋮                       ；计算脉宽和处理
```

【例 4】 利用定时 / 计数器 0 或定时 / 计数器 1 的 Tx 端口改造成外部中断源输入端口的应用设计。

在某些应用系统中常会出现原有的两个外部中断源 INT0 和 INT1 不够用，而定时 / 计数器有多余，则可将 Tx 用于增加的外部中断源。现选择定时 / 计数器 1 为对外部事件计数模式工作方式 2（自动再装入），设置计数初值为 FFH，则 T1 端口输入一个负跳变脉冲，计数器即回 0 溢出，置位对应的中断请求标志位 TF1 为 1，向主机请求中断处理，从而达到了增加一个外部中断源的目的。应用定时 / 计数器 1（T1）的中断矢量转入中断服务程序处理。其程序示例如下：

（1 ）主程序段：

```
    ORG   0000H
    AJMP  MAIN              ；转主程序
    ORG   001BH
    LJMP  INTER             ；转 T1 中断服务程序
     ⋮
    ORG   0100              ；主程序入口
MAIN：…
     ⋮
    MOV   SP，#60H          ；设置堆栈区
    MOV   TMOD，#60H        ；设置定时 / 计数器 1，计数方式 2
    MOV   TL1，#0FFH        ；设置计数常数
    MOV   TH1，#0FFH
    SETB  EA                ；开中断
    SETB  ET1               ；开定时 / 计数器 1 中断
    SETB  TR1               ；启动定时 / 计数器 1 计数
     ⋮
```

（2 ）中断服务程序（具体处理程序略）

```
            ORG   1000H
INTER：     PUSH  A             ;
                                } 现场入栈保护
            PUSH  DPL           ;
            PUSH  DPH           ;
             ⋮
```

```
        ⋮                          } 中断处理主体程序
        ⋮
    POP   DPH
    POP   DPL                      } 现场出栈复原
    POP   A
    RETI                           ；返回
```

这是中断服务程序的基本格式。

【例 5】 某应用系统需通过 P1.0 和 P1.1 分别输出周期为 200 μs 和 400 μs 的方波。为此，系统选用定时器 / 计数器 0（T0），定时方式 3，主频为 6MHz，TP=2 μs，经计算得定时常数为 9CH 和 38H。

本例程序段编制如下：

（1）初始化程序段

```
        ⋮
PLT0:MOV   TMOD,#03H           ；设置 T0 定时方式 3
     MOV   TL0，#9CH           ；设置 TL0 初值
     MOV   TH0，#38H           ；设置 TH0 初值
     SETB  EA                  ；
     SETB  ET0                 ；} 开中断
     SETB  ET1                 ；
     SETB  TR0                 ；启动
     SETB  TR1                 ；启动
      ⋮
```

(2)中断服务程序段

1)

```
INT0P：  ⋮
         ⋮
    MOV  TL0，#9CH             ；重新设置初值
    CPL  P1.0                  ；对 P1.0 输出信号取反
         ⋮
    RETI                       ；返回
```

2)

```
INT1P    ⋮
         ⋮
    MOV  TH0，#38H             ；重新设置初值
    CPL  P1.1                  ；对 P1.1 输出信号取反
         ⋮
    RETI                       ；返回
```

在实际应用中应注意的问题如下。

（1）定时 / 计数器的实时性

定时 / 计数器启动计数后，当计满回 0 溢出向主机请求中断处理，由内部硬件自动进行。但从回 0 溢出请求中断到主机响应中断并作出处理存在时间延迟,且这种延时随中断请求时的现场环境的不同而不同,一般需延时 3 个机器周期以上，这就给实时处理带来误差。大多数应用场合可忽略不计，但对某些要求实时性苛刻的场合，应采用补偿措施。

这种由中断响应引起的时间延时，对定时 / 计数器工作于方式 0 或 1 而言有两种含义：一是由于中断响应延时而引起的实时处理的误差；二是如需多次且连续不间断地定时 / 计数，由于中断响应延时，则在中断服务程序中再置计数初值时已延误了若干个计数值而引起误差，特别是用于定时就更明显。

例如选用定时方式 1 设置系统时钟，由于上述原因就会产生实时误差。这种场合应采用动态补偿办法以减少系统始终误差。所谓动态补偿，即在中断服务程序中对 THx、TLx 重新置计数初值时，应将 THx、TLx 从回 0 溢出又重新从 0 开始继续计数的值读出，并补偿到原计数初值中去进行重新设置。可考虑如下补偿方法：

```
    ⋮
CLR  EA              ；禁止中断
MOV  A，TLx          ；读 TLx 中已计数值
ADD  A，#LOW         ；LOW 为原低字节计数初值
MOV  TLx，A          ；设置低字节计数初值
MOV  A，#HIGH        ；原高字节计数初值送 A
ADDC A，THx          ；高字节计数初值补偿
MOV  THx，A          ；置高字节计数初值
SETB EA              ；开中断
    ⋮
```

（2）动态读取运行中的计数值

在动态读取运行中的定时 / 计数器的计数值时，如果不加注意，就可能出错。这是因为不可能在同一时刻同时读取 THx 和 TLx 中的计数值。比如，先读 TLx 后读 THx，因为定时 / 计数器处于运行状态，在读 TLx 时尚未产生向 THx 进位，而在读 THx 前已产生进位，这时读得的 THx 就不对了；同样，先读 THx 后读 TLx 也可能出错。

一种可避免读错的方法是：先读 THx，后读 TLx，将两次读得的 THx 进行比较；若两次读得的值相等，则可确定读的值是正确的，否则重复上述过程，重复读得的值一般不会再错。此法的软件编程如下：

```
RDTM： MOV  A，THx                ；读取 THx 存 A 中
       MOV  R0，TLx               ；读取 TLx 存 R0 中
       CJNE A，THx，RDTM          ；比较两次 THx 值，若相等，则读得的值正
                                  ；确，程序往下执行，否则重读
       MOV  R1，A                 ；将 THx 存于 R1 中
        ⋮
```

6.3 用定时器 1 做波特率发生器

```
;/* ---  STC International Limited ------------------------- */
;/* ---  宏晶科技 姚永平  设计 2006/1/6   V1.0 -------------- */
;/* ---  演示 STC12C5410AD 系列 MCU 定时器 1 作波特率发生器功能 - */
;/* ---  演示 STC12C2052AD 系列 MCU 定时器 1 作波特率发生器功能 - */
;/* ---  Mobile: 13922805190 ------------------------------- */
;/* ---  Fax: 0755-82944243 -------------------------------- */
;/* ---  Tel: 0755-82948409 -------------------------------- */
;/* ---  Web: www.mcu-memory.com --------------------------- */
;本演示程序在宏晶的 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过
;如果要在程序中使用或在文章中引用该程序,请在程序中或文章中注明使用了宏晶科技的资料及程序
;-------------------------------------------------------------------------------
;     本程序演示 STC12C2052AD、STC12C5410AD 系列单片机用定时器 1 作 RS-232 通信
;波特率发生器的使用方法，有关波特率自动重装数的计算请查看程序后面的内容
;     本程序同时演示 STC89C51RC/STC89C52RC/STC89C53RC/STC89C54RD+/STC89C58RD+/
;STC89C516RD+ 系列单片机用定时器 1 作 RS-232 通信波特率发生器的使用方法。
;     STC12C2052AD、STC12C5410AD 系列是 " 一个时钟 / 机器周期 " 的 8051 单片机。它
;的定时器 0、定时器 1 有两种计数速率，一种是 12T 模式：每 12 个时钟加 1，与普通的
;8051 单片机相同；另一种是 1T 模式：每个时钟加 1，是普通 8051 单片机的 12 倍。
;     STC89C51RC/RD+ 系列是 "12 个时钟 / 机器周期 " 的 8051 单片机，与普通的 8051 单片
; 机相同。
;     STC12C2052AD、STC12C5410AD 系列的单片机，定时器 0、定时器 1 的计数速率由
;特殊功能寄存器 AUXR 的 bit7，bit6 决定，bit7 的符号是 T0x12，如果 T0x12=1，
;定时器 0 就工作在 1T 模式。bit6 的符号是 T1x12，如果 T1x12=1，定时器 1 就工作在
;1T 模式。有关详情请参考 STC12C5410AD 系列单片机器件手册(中文应用指南)。

;使用方法:
;  1. 修改程序，改变波特率参数或改变定时器 1 的计数速率(1T 模式 /12T 模式)
;  2. 汇编程序，将代码下载到单片机中
;  3. 调整串口调试助手的波特率与单片机的波特率相同，并打开调试助手的串口。STC
;     下载程序 STC-ISP.exe 版本 3.2 以上有串口调试助手功能。
;  4. 打开单片机电源，可以在串口调试助手的接收区看到单片机发出的数据
;  5. 用串口调试助手发送单个字节到单片机，单片机收到后会立即回发到串口调试助手
;  6. 反复步骤 1-5，检验波特率参数是否正确，特别要观察定时器 1 工作在 1T 模式
;     的波特率。例如，先设置定时器 1 工作在 12T 模式，设置波特率为 9600，执行
;     步骤 2-5，检验波特率参数是否正确。然后仅仅将定时器 1 的计数速率改成
;     1T 模式，执行步骤 2-5，就会发现本程序的波特率变成了 115200，波特率是
;     12T 模式的 12 倍。
;
;-------------------------------------------------------------------------------
```

```
;定义 STC12C5410AD 系列 MCU 特殊功能寄存器
AUXR    EQU  8EH
;------------------------------------------------------------------------------
;定义波特率自动重装数
;******************************************************************************
;以下是 Fosc = 22.1184MHz，1T 模式，SMOD=1 时，计算出的自动重装数和波特率
;RELOAD_COUNT EQU 0FFH      ;Baud=1,382,400 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=691,200 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=460,800 bps
;RELOAD_COUNT EQU 0FCH      ;Baud=345,600 bps
;RELOAD_COUNT EQU 0FBH      ;Baud=276,480 bps
;RELOAD_COUNT EQU 0FAH      ;Baud=230,400 bps
;RELOAD_COUNT EQU 0F4H      ;Baud=115,200 bps
;RELOAD_COUNT EQU 0E8H      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0DCH      ;Baud=38,400 bps
;RELOAD_COUNT EQU 0B8H      ;Baud=19,200 bps
;RELOAD_COUNT EQU 70H      ;Baud=9,600 bps
;以上是 Fosc = 22.1184MHz，1T 模式，SMOD=1 时，计算出的自动重装数和波特率
;******************************************************************************

;******************************************************************************
;以下是 Fosc = 1.8432MHz，1T 模式，SMOD=1 时，计算出的自动重装数和波特率
;RELOAD_COUNT EQU 0FFH      ;Baud=115,200 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=38,400 bps
;RELOAD_COUNT EQU 0FCH      ;Baud=28,800 bps
;RELOAD_COUNT EQU 0FAH      ;Baud=19,200 bps
;RELOAD_COUNT EQU 0F4H      ;Baud=9,600 bps
;RELOAD_COUNT EQU 0E8H      ;Baud=4,800 bps
;RELOAD_COUNT EQU 0D0H      ;Baud=2,400 bps
;RELOAD_COUNT EQU 0A0H      ;Baud=1,200 bps
;以上是 Fosc = 1.8432MHz，1T 模式，SMOD=1 时，计算出的自动重装数和波特率
;******************************************************************************

;******************************************************************************
;以下是 Fosc = 18.432MHz，1T 模式，SMOD=1 时，计算出的自动重装数和波特率
;RELOAD_COUNT EQU 0FFH      ;Baud=1,152,000 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=576,000 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=288,000 bps
;RELOAD_COUNT EQU 0FCH      ;Baud=144,000 bps
;RELOAD_COUNT EQU 0F6H      ;Baud=115,200 bps
;RELOAD_COUNT EQU 0ECH      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0E2H      ;Baud=38,400 bps
;RELOAD_COUNT EQU 0D8H      ;Baud=28,800 bps
;RELOAD_COUNT EQU 0C4H      ;Baud=19,200 bps
;RELOAD_COUNT EQU 088H      ;Baud=9,600 bps
;以上是 Fosc = 18.432MHz，1T 模式，SMOD=1 时，计算出的自动重装数和波特率
;******************************************************************************
```

```
;*******************************************************************************
;以下是 Fosc = 18.432MHz，1T 模式，SMOD=0 时，计算出的自动重装数和波特率

;RELOAD_COUNT EQU 0FFH      ;Baud=576,000 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=288,000 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=144,000 bps
;RELOAD_COUNT EQU 0FCH      ;Baud=115,200 bps
;RELOAD_COUNT EQU 0F6H      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0ECH      ;Baud=38,400 bps
;RELOAD_COUNT EQU 0E2H      ;Baud=28,800 bps
;RELOAD_COUNT EQU 0D8H      ;Baud=19,200 bps
;RELOAD_COUNT EQU 0C4H      ;Baud=9,600 bps
;RELOAD_COUNT EQU 088H      ;Baud=4,800 bps

;以上是 Fosc = 18.432MHz，1T 模式，SMOD=0 时，计算出的自动重装数和波特率
;*******************************************************************************

;*******************************************************************************
;以下是 Fosc = 18.432MHz，12T 模式，SMOD=0 时，计算出的自动重装数和波特率

RELOAD_COUNT EQU 0FBH      ;Baud=9,600 bps
;RELOAD_COUNT EQU 0F6H      ;Baud=4,800 bps
;RELOAD_COUNT EQU 0ECH      ;Baud=2,400 bps
;RELOAD_COUNT EQU 0D8H      ;Baud=1,200 bps

;以上是 Fosc = 18.432MHz，12T 模式，SMOD=0 时，计算出的自动重装数和波特率
;*******************************************************************************

;*******************************************************************************
;以下是 Fosc = 18.432MHz，12T 模式，SMOD=1 时，计算出的自动重装数和波特率

;RELOAD_COUNT EQU 0FBH      ;Baud=19,200 bps
;RELOAD_COUNT EQU 0F6H      ;Baud=9,600 bps
;RELOAD_COUNT EQU 0ECH      ;Baud=4,800 bps
;RELOAD_COUNT EQU 0D8H      ;Baud=2,400 bps
;RELOAD_COUNT EQU 0B0H      ;Baud=1,200 bps

;以上是 Fosc = 18.432MHz，12T 模式，SMOD=1 时，计算出的自动重装数和波特率
;*******************************************************************************
```

```
;*******************************************************************************
;以下是 Fosc = 11.0592MHz，12T 模式，SMOD=0 时，计算出的自动重装数和波特率

;RELOAD_COUNT EQU 0FFH      ;Baud=28,800 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=14,400 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=9,600 bps
;RELOAD_COUNT EQU 0FAH      ;Baud=4,800 bps
;RELOAD_COUNT EQU 0F4H      ;Baud=2,400 bps
;RELOAD_COUNT EQU 0E8H      ;Baud=1,200 bps

;以上是 Fosc = 11.0592MHz，12T 模式，SMOD=0 时，计算出的自动重装数和波特率
;*******************************************************************************

;*******************************************************************************
;以下是 Fosc = 11.0592MHz，12T 模式，SMOD=1 时，计算出的自动重装数和波特率

;RELOAD_COUNT EQU 0FFH      ;Baud=57,600 bps
;RELOAD_COUNT EQU 0FEH      ;Baud=28,800 bps
;RELOAD_COUNT EQU 0FDH      ;Baud=14,400 bps
;RELOAD_COUNT EQU 0FAH      ;Baud=9,600 bps
;RELOAD_COUNT EQU 0F4H      ;Baud=4,800 bps
;RELOAD_COUNT EQU 0E8H      ;Baud=2,400 bps
;RELOAD_COUNT EQU 0D0H      ;Baud=1,200 bps

;以上是 Fosc = 11.0592MHz，12T 模式，SMOD=1 时，计算出的自动重装数和波特率
;*******************************************************************************

;定义指示灯
LED_MCU_START  EQU  P1.7            ;MCU 工作指示灯
;-------------------------------------------------------------------------------
     ORG   0000H
    AJMP  MAIN
;-------------------------------------------------------------------------------
    ORG   0023H
    AJMP  UART_Interrupt            ;RS232 串口中断服务程序
    NOP
    NOP
;-------------------------------------------------------------------------------
MAIN:
    MOV   SP, #7FH                  ;设置堆栈指针
    CLR   LED_MCU_START             ;点亮 MCU 工作指示灯
    ACALL Initial_UART              ;初始化串口
    MOV   R0, #30H                  ;30H = 可打印字符 '0' 的 ASCII 码
    MOV   R2, #10                   ;发送 10 个字符 '0123456789'
```

```
LOOP:
    MOV   A，R0
    ACALL Send_One_Byte              ;发送一个字节,可将 PC 串口调试助手设置成字符显示
    ;如果是字符显示，显示为 0123456789，
    ;如设置成 16 进制显示，显示 30 31 32 33 34 35 36 37 38 39
    INC   R0
    DJNZ  R2，LOOP
MAIN_WAIT:
    SJMP  MAIN_WAIT                  ;跳转到本行，无限循环
;--------------------------------------------------------------------------------
UART_Interrupt:                      ;串口中断服务程序
    JB    RI，Is_UART_Receive
    CLR   TI                         ;清零串口发送中断标志
    RETI                             ;发送时使用的是查询方式，不使用中断
Is_UART_Receive:
    CLR   RI
    PUSH  ACC
    MOV   A，SBUF                    ;取接收到的字节
    ACALL Send_One_Byte              ;回发收到的字节
    POP   ACC
    RETI
;--------------------------------------------------------------------------------
Initial_UART:                        ;初始化串口
; SCON  Bit:   7       6     5     4     3     2     1    0
;            SM0/FE   SM1   SM2   REN   TB8   RB8   TI   RI
    MOV   SCON，#50H                 ; 0101,0000 8 位可变波特率，无奇偶校验

    MOV   TMOD，#21H                 ;设置定时器 1 为 8 位自动重装计数器
    MOV   TH1，#RELOAD_COUNT         ;设置定时器 1 自动重装数
    MOV   TL1，#RELOAD_COUNT

;--------------------------------------
;    ORL   PCON，#80H                ;若本行有效，波特率可以加倍
;--------------------------------------
;以下两行指令只能有一行有效
;    ORL   AUXR，#01000000B           ;定时器 1 工作在 1T 模式，波特率可以快 12 倍
     ANL   AUXR，#10111111B           ;定时器 1 工作在 12T 模式，与普通的 8051 相同
;以上两行指令只能有一行有效
;--------------------------------------
    SETB  TR1                        ;启动定时器 1
    SETB  ES
    SETB  EA
    RET
```

```
;-------------------------------------------------------------------------------
;入口参数: A = 要发送的字节
Send_One_Byte:                          ;发送一个字节
    CLR   ES
    CLR   TI                            ;清零串口发送中断标志
    MOV   SBUF, A
Wait_Send_Finish:
    JNB   TI, Wait_Send_Finish          ;等待发送完毕
    CLR   TI                            ;清零串口发送中断标志
    SETB  ES
    RET
;-------------------------------------------------------------------------------
    END
;-------------------------------------------------------------------------------
;计算自动重装数 RELOAD (SMOD = 0, SMOD 是 PCON 特殊功能寄存器的最高位):
;  1. 计算 RELOAD (以下是 SMOD = 0 时的计算公式)
;
;     a) 12T 模式的计算公式： RELOAD = 256 - INT(Fosc/Baud0/32/12 + 0.5)
;     b)  1T 模式的计算公式： RELOAD = 256 - INT(Fosc/Baud0/32 + 0.5)
;
;    式中: INT() 表示取整运算即舍去小数, 在式中加 0.5 可以达到四舍五入的目的
;          Fosc = 晶振频率
;          Baud0 = 标准波特率
;
;  2. 计算用 RELOAD 产生的波特率:
;       a)   Baud = Fosc/(256 - RELOAD)/32/12       12T 模式
;       b)   Baud = Fosc/(256 - RELOAD)/32           1T 模式
;
;  3. 计算误差
;          error = (Baud - Baud0)/Baud0 * 100%
;  4. 如果误差绝对值 > 4.5% 要更换波特率或者更换晶体频率, 重复步骤 1-4
;
;
;例: Fosc = 22.1184MHz, Baud0 = 57600 (12T 模式)
; 1. RELOAD = 256 - INT( 22118400/57600/32/12 + 0.5)
;           = 256 - INT( 1.5 )
;           = 256 - 1
;           = 255
;           = 0FFH
; 2. Baud = 22118400/(256-255)/32/12
;         = 57600
; 3. 误差等于零
```

```
;例: Fosc = 18.432MHz, Baud0 = 57600 (12T 模式)
; 1. RELOAD = 256 - INT( 18432000/57600/32/12 + 0.5)
;           = 256 - INT( 0.833 + 0.5 )
;           = 256 - INT( 1.333 )
;           = 256 - 1
;           = 255
;           = 0FFH
; 2. Baud = 18432000/(256-255)/32/12
;         = 48000
; 3. error = (48000 - 57600)/57600 * 100%
;          = -16.66%
; 4. 误差很大，要更换波特率或者更换晶体频率，重新计算请见下一例

;例: Fosc = 18.432MHz, Baud0 = 9600 (12T 模式)
; 1. RELOAD = 256 - INT( 18432000/9600/32/12 + 0.5)
;           = 256 - INT( 5.5 )
;           = 256 - 5
;           = 251
;           = 0FBH
; 2. Baud = 18432000/(256-251)/32/12
;         = 9600
; 3. 一目了然，误差等于零

;例: Fosc = 2.000MHz, Baud = 4800 (1T 模式)
; 1. RELOAD = 256 - INT( 2000000/4800/32 + 0.5)
;           = 256 - INT( 13.02 + 0.5 )
;           = 256 - INT( 13.52 )
;           = 256 - 13
;           = 243
;           = 0F3H
; 2. Baud = 2000000/(256-243)/32
;         = 4808
; 3. error = 0.16%
;-------------------------------------------------------------------------------
```

第七章 STC12系列单片机的A/D转换

7.1 A/D转换寄存器

STC12C5410AD系列带A/D转换的单片机的A/D转换口在P1口(P1.7-P1.0)，有8路10位高速A/D转换器,STC12C2052AD系列是8位精度的A/D，速度均可达到100KHz(10万次/秒)。8路电压输入型A/D，可做温度检测、电池电压检测、按键扫描、频谱检测等。上电复位后P1口为弱上拉型I/O口，用户可以通过软件设置将8路中的任何一路设置为A/D转换，不需作为A/D使用的口可继续作为I/O口使用。

需作为A/D使用的口需先将其设置为开漏模式或高阻输入,在P1M0、P1M1寄存器中对相应的位进行设置。

P1M0【7：0】 地址：91h	P1M1【7：0】 地址：92h	I/O口模式（P1.x 如做A/D使用，需先将其设置成开漏或高阻输入）
0	0	准双向口（传统8051 I/O口模式）， 灌电流可达20mA，拉电流为230μA
0	1	推挽输出（强上拉输出，可达20mA，尽量少用）
1	0	仅为输入（高阻），如果该I/O口需作为A/D使用，可选此模式
1	1	开漏(Open Drain)，如果该I/O口需作为A/D使用，可选此模式

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ADC_CONTR	C5h	A/D转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0xx0,0000
ADC_DATA	C6h	A/D转换结果寄存器,高8位	-	-	-	-	-	-	-	-	xxxx,xxxx
ADC_LOW2	BEh	A/D转换结果寄存器,低2位	-	-	-	-	-	-	-	-	xxxx,xxxx

ADC_CONTR 特殊功能寄存器： A/D转换控制特殊功能寄存器

A/D转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0xx0,0000

CHS2 / CHS1 / CHS0：模拟输入通道选择，CHS2 / CHS1 / CHS0

CHS2	CHS1	CHS0	Analog Channel Select 模拟输入通道选择
0	0	0	选择P1.0作为A/D输入来用
0	0	1	选择P1.1作为A/D输入来用
0	1	0	选择P1.2作为A/D输入来用
0	1	1	选择P1.3作为A/D输入来用
1	0	0	选择P1.4作为A/D输入来用
1	0	1	选择P1.5作为A/D输入来用
1	1	0	选择P1.6作为A/D输入来用
1	1	1	选择P1.7作为A/D输入来用

ADC_START： 模数转换器(ADC)转换启动控制位，设置为“1”时，开始转换,转换结束后为0。

ADC_FLAG： 模数转换器转换结束标志位,当A/D转换完成后，ADC_FLAG = 1，要由软件清0。

不管是A/D转换完成后由该位申请产生中断，还是由软件查询该标志位A/D转换是否结束，当A/D转换完成后，ADC_FLAG = 1，一定要软件清0。

SPEED1，SPEED0：模数转换器转换速度控制位

SPEED1	SPEED0	A/D转换所需时间
1	1	270个时钟周期转换一次，CPU工作频率27MHz时，A/D转换速度约100KHz
1	0	540个时钟周期转换一次
0	1	810个时钟周期转换一次
0	0	1080个时钟周期转换一次

ADC_POWER: ADC 电源控制位。

0：关闭 ADC 电源；1：打开 A/D 转换器电源.建议进入空闲模式前，将 ADC 电源关闭，ADC_POWER =0.

启动 AD 转换前一定要确认 AD 电源已打开，AD 转换结束后关闭 AD 电源可降低功耗，也可不关闭。

初次打开内部 A/D 转换模拟电源，需适当延时，等内部模拟电源稳定后，再启动 A/D 转换

建议启动 A/D 转换后，在 A/D 转换结束之前，不改变任何 I/O 口的状态，有利于高精度 A/D 转换

ADC_DATA / ADC_LOW2 特殊功能寄存器: A/D 转换结果特殊功能寄存器

ADC_DATA	C6h	A/D 转换结果寄存器,全部8位有效，为10位A/D转换结果的高8位	-	-	-	-	-	-	-	-	xxxx,xxxx
ADC_LOW2	BEh	A/D 转换结果寄存器，只有低2位有效，为10位A/D转换结果的低2位	x	x	x	x	x	x	-	-	xxxx,xxxx

模拟/数字转换结果计算公式如下：结果 (ADC_DATA[7:0],ADC_LOW2[1:0]) = 1024 x Vin / Vcc

Vin 为模拟输入通道输入电压，Vcc 为单片机实际工作电压，用单片机工作电压作为模拟参考电压。

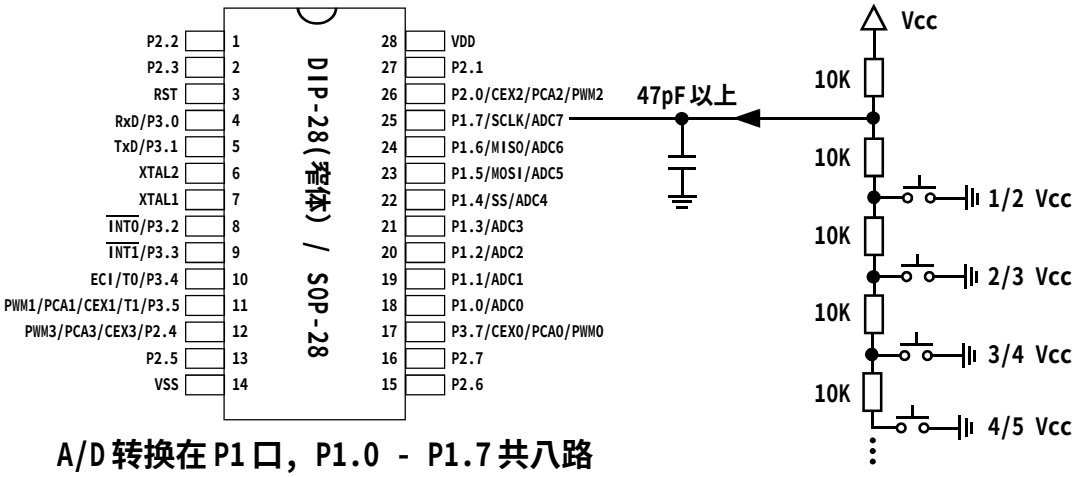
取 ADC_DATA 的 8 位为 ADC 转换的高 8 位,取 ADC_LOW2 的低 2 位为 ADC 转换的低 2 位,则为 10 位精度。

如果舍弃 ADC_LOW2 的低 2 位,只用 ADC_DATA 寄存器的 8 位,则 A/D 转换结果为 8 位精度。

结果 ADC_DATA[7:0] = 256 x Vin / Vcc

STC12C2052AD 系列单片机 A/D 转换精度只有 8 位，固无 ADC_LOW2 寄存器。

7.2 A/D 转换典型应用线路，按键扫描



A/D 转换在 P1 口，P1.0 - P1.7 共八路

7.3 A/D 转换模块的参考电压源

STC12C5410AD 和 STC12C2052AD 系列单片机的参考电压源是输入工作电压 Vcc，所以一般不用外接参考电压源。如 7805 的输出电压是 5V，但实际电压可能是 4.88V 到 4.96V，用户需要精度比较高的话，可在出厂时将实际测出的工作电压值记录在单片机内部的 EEPROM 里面，以供计算。

如果有些用户的 Vcc 不固定，如电池供电，电池电压在 5.3V-4.2V 之间漂移，则 Vcc 不固定，就需要在 8 路 A/D 转换的一个通道外接一个稳定的参考电压源，来计算出此时的工作电压 Vcc，再计算出其他几路 A/D 转换通道的电压。

如可在 ADC 转换通道的第七通道外接一个 1.25V（或 1V，或...）的基准参考电压源，由此求出此时的工作电压 Vcc，再计算出其它几路 A/D 转换通道的电压。

7.4 一个完整的 A/D 转换测试程序

```
;/* ---  STC International Limited ---------------- */
;/* ---  宏晶科技 姚永平  设计 2006/1/6   V1.0 ----- */
;/* ---  演示 STC12C5410AD 系列 MCU 的 A/D 转换功能 - */
;/* ---  演示 STC12C2052AD 系列 MCU 的 A/D 转换功能 - */
;/* ---  Mobile: 13922805190 ---------------------- */
;/* ---  Fax: 0755-82944243 ----------------------- */
;/* ---  Tel: 0755-82948409 ----------------------- */
;/* ---  Web: www.mcu-memory.com ------------------ */
;如果要在程序中使用或在文章中引用该程序,请在程序中或文章中注明使用了宏晶科技的资料及程序

;ADC DEMO_5410_ASM.ASM 汇编程序演示 STC12C5410AD 系列 MCU 的 A/D 转换功能。
;本程序用宏晶的 STC-ISP Ver 3.0A.PCB 的下载编程工具测试通过,相关的 A/D 转换结果在 P1 口上显示
;转换结果也以 16 进制形式输出到串行口，可以用串行口调试程序观察输出结果。
;时钟 18.432MHz，波特率 = 9600。
;转换结果也在 P1 口利用 LED 显示出来，方便观察。

LED_MCU_START    EQU  P3.7
ADC_CONTR  EQU  0C5H    ;A/D 转换寄存器
ADC_DATA   EQU  0C6H    ;A/D 转换结果寄存器，为 10 位 A/D 转换结果的高 8 位
;ADC_LOW2    EQU 0BEH    ;A/D 转换结果寄存器，低 2 位有效，为 10 位 A/D 转换结果的低 2 位
                         ;如果不用 ADC_LOW2 的低 2 位,只用 ADC_DATA 的 8 位,则为 8 位 A/D 转换
P1M0       EQU  91H     ;P1 口模式选择寄存器 0
P1M1       EQU  92H     ;P1 口模式选择寄存器 1

ADC_Power_On_Speed_Channel_0 EQU 11100000B     ;P1.0 作为 A/D 输入
ADC_Power_On_Speed_Channel_1 EQU 11100001B     ;P1.1 作为 A/D 输入
ADC_Power_On_Speed_Channel_2 EQU 11100010B     ;P1.2 作为 A/D 输入
ADC_Power_On_Speed_Channel_3 EQU 11100011B     ;P1.3 作为 A/D 输入
ADC_Power_On_Speed_Channel_4 EQU 11100100B     ;P1.4 作为 A/D 输入
ADC_Power_On_Speed_Channel_5 EQU 11100101B     ;P1.5 作为 A/D 输入
ADC_Power_On_Speed_Channel_6 EQU 11100110B     ;P1.6 作为 A/D 输入
ADC_Power_On_Speed_Channel_7 EQU 11100111B     ;P1.7 作为 A/D 输入

;------------------------------------------------
;定义变量
ADC_Channel_0_Result      EQU 30H     ;0 通道 A/D 转换结果
ADC_Channel_1_Result      EQU 31H     ;1 通道 A/D 转换结果
ADC_Channel_2_Result      EQU 32H     ;2 通道 A/D 转换结果
ADC_Channel_3_Result      EQU 33H     ;3 通道 A/D 转换结果
ADC_Channel_4_Result      EQU 34H     ;4 通道 A/D 转换结果
ADC_Channel_5_Result      EQU 35H     ;5 通道 A/D 转换结果
ADC_Channel_6_Result      EQU 36H     ;6 通道 A/D 转换结果
ADC_Channel_7_Result      EQU 37H     ;7 通道 A/D 转换结果
```

```
;------------------------------------------------
    ORG   0000H
    LJMP  MAIN

    ORG   0050H
MAIN:
    CLR   LED_MCU_START             ;MCU 工作指示灯 LED_MCU_START  EQU P3.7
    MOV   SP, #7FH                  ;设置堆栈

    ACALL Initiate_RS232            ;初始化串口

    ACALL ADC_Power_On              ;开 ADC 电源，第一次使用时要打开内部模拟电源
                                    ;开 ADC 电源，可适当加延时，1mS 以内就足够了
    ACALL Set_P12_Open_Drain        ;设置 P1.2 为开漏
    ACALL Set_ADC_Channel_2         ;设置 P1.2 作为 A/D 转换通道

    ACALL Get_AD_Result             ;测量电压并且取 A/D 转换结果
    ACALL Send_AD_Result            ;发送转换结果到 PC 机

    ACALL Set_P12_Normal_IO         ;设置 P1.2 为普通 IO
    MOV   A, ADC_Channel_2_Result   ;用 P1 口显示 A/D 转换结果
    CPL   A
    MOV   P1, A

Wait_Loop:
    SJMP  Wait_Loop                 ;停机

;------------------------------------------------------------
;------------------------------------------------------------
;------------------------------------------------------------
;------------------------------------------------------------
Initiate_RS232:                     ;串口初始化
    CLR   ES                        ;禁止串口中断
    MOV   TMOD, #20H                ;设置 T1 为波特率发生器
    MOV   SCON, #50H                ;0101,0000 8 位数据位，无奇偶校验
    MOV   TH1, #0FBH                ;18.432MHz 晶振，波特率 = 9600
    MOV   TL1, #0FBH

    SETB  TR1                       ;启动 T1
    RET
;------------------------------------------------------------
Send_Byte:
    CLR   TI
    MOV   SBUF, A
Send_Byte_Wait_Finish:
    JNB   TI, Send_Byte_Wait_Finish
    CLR   TI
    RET
```

```
;-------------------------------------------------------------
ADC_Power_On:
    PUSH  ACC
    ORL   ADC_CONTR, #80H            ;开 A/D 转换电源
    MOV   A, #20H
    ACALL Delay                      ;开 A/D 转换电源后要加延时，1mS 以内就足够了
    POP   ACC
    RET

;-------------------------------------------------------------
;设置 P1.2,设置 A/D 通道所在的 I/O 为开漏模式
Set_P12_Open_Drain:
    PUSH  ACC
    MOV   A, #00000100B
    ORL   P1M0, A
    ORL   P1M1, A
    POP   ACC
    RET

;-------------------------------------------------------------
;设置 P1.2 为普通 IO
Set_P12_Normal_IO:
    PUSH  ACC
    MOV   A, #11111011B
    ANL   P1M0, A
    ANL   P1M1, A
    POP   ACC
    RET

;-------------------------------------------------------------
Set_ADC_Channel_2:
    MOV   ADC_CONTR, #ADC_Power_On_Speed_Channel_2
                     ;选择 P1.2 作为 A/D 转换通道
    MOV   A, #05H    ;更换 A/D 转换通道后要适当延时，使输入电压稳定
                     ;以后如果不更换 A/D 转换通道的话，不需要加延时
    ACALL Delay      ;切换 A/D 转换通道,加延时 20uS～200uS 就可以了,与输入电压源的内阻有关
                     ;如果输入电压信号源的内阻在 10K 以下,可不加延时
    RET

;-------------------------------------------------------------

Send_AD_Result:
    PUSH  ACC
    MOV   A, ADC_Channel_2_Result    ;取 AD 转换结果
    ACALL Send_Byte                  ;发送转换结果到 PC 机
    POP   ACC
    RET
```

```
;-------------------------------------------------------
Get_AD_Result:
    PUSH  ACC                       ;入栈保护
    MOV   ADC_DATA, #0
    ORL   ADC_CONTR, #00001000B     ;启动 AD 转换

Wait_AD_Finishe:
    MOV   A, #00010000B             ;判断 AD 转换是否完成
    ANL   A, ADC_CONTR
    JZ    Wait_AD_Finishe           ;AD 转换尚未完成，继续等待

    ANL   ADC_CONTR, #11100111B     ;清 0 ADC_FLAG, ADC_START 位，停止 A/D 转换

    MOV   A, ADC_DATA
    MOV   ADC_Channel_2_Result, A   ;保存 AD 转换结果
    POP   ACC
    RET
;-------------------------------------------------------------
Delay:
    PUSH  02                        ;将寄存器组 0 的 R2 入栈
    PUSH  03                        ;将寄存器组 0 的 R3 入栈
    PUSH  04                        ;将寄存器组 0 的 R4 入栈
    MOV   R4, A

Delay_Loop0:
    MOV   R3, #200              ;2 CLOCK  ----------------------+
Delay_Loop1:                    ;                               |
    MOV   R2, #249              ;2 CLOCK  ------+               |
Delay_Loop:                     ;               | 1002 CLOCK   |200406 CLOCK
    DJNZ  R2, Delay_Loop        ;4 CLOCK        |               |
    DJNZ  R3, Delay_Loop1       ;4 CLOCK  ------+               |
    DJNZ  R4, Delay_Loop0       ;4 CLOCK  ----------------------+

    POP   04
    POP   03
    POP   02
    RET
;-------------------------------------------------------------
    END
```

第八章 STC12 系列单片机的 PCA/PWM 应用

8.1 PCA/PWM 寄存器列表

STC12C5410AD 及 STC12C2052AD 系列 1T 8051 单片机 PCA/PWM 特殊功能寄存器表 PCA/PWM SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CCON	D8h	PCA Control Register	CF	CR	-	-	CCF3	CCF2	CCF1	CCF0	00xx,0000
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CCAPM2	DCh	PCA Module 2 Mode Register	-	ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2	x000,0000
CCAPM3	DDh	PCA Module 3 Mode Register	-	ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3	x000,0000
CL	E9h	PCA Base Timer Low									0000,0000
CH	F9h	PCA Base Timer High									0000,0000
CCAP0L	EAh	PCA Module-0 Capture Register Low									0000,0000
CCAP0H	FAh	PCA Module-0 Capture Register High									0000,0000
CCAP1L	EBh	PCA Module-1 Capture Register Low									0000,0000
CCAP1H	FBh	PCA Module-1 Capture Register High									0000,0000
CCAP2L	ECh	PCA Module-2 Capture Register Low									0000,0000
CCAP2H	FCh	PCA Module-2 Capture Register High									0000,0000
CCAP3L	EDh	PCA Module-3 Capture Register Low									0000,0000
CCAP3H	FDh	PCA Module-3 Capture Register High									0000,0000
PCA_PWM0	F2h	PCA PWM Mode Auxiliary Register 0	-	-	-	-	-	-	EPC0H	EPC0L	xxxx,xx00
PCA_PWM1	F3h	PCA PWM Mode Auxiliary Register 1	-	-	-	-	-	-	EPC1H	EPC1L	xxxx,xx00
PCA_PWM2	F4h	PCA PWM Mode Auxiliary Register 2	-	-	-	-	-	-	EPC2H	EPC2L	xxxx,xx00
PCA_PWM3	F5h	PCA PWM Mode Auxiliary Register 3	-	-	-	-	-	-	EPC3H	EPC3L	xxxx,xx00

CMOD - PCA 模式 寄存器的位分配 （地址：D9H）

位	7	6	5	4	3	2	1	0
符 号	CIDL	-	-	-	-	CPS1	CPS0	ECF

CMOD - PCA 模式 寄存器的位描述 （地址：D9H）

位	符号	描述
7	CIDL	计数器阵列空闲控制：CIDL=0时，空闲模式下PCA计数器继续工作。CIDL＝1时，空闲模式下PCA计数器停止工作。
6 - 3	-	保留为将来之用。
2 - 1	CPS1,CPS0	PCA计数脉冲选择（见下表）。
0	ECF	PCA计数溢出中断使能：ECF=1时，使能寄存器CCON CF位的中断。ECF=0时，禁止该功能。

CMOD - PCA 计数器阵列的计数脉冲选择 （地址：D9H）

CPS1	CPS0	选择PCA时钟源输入
0	0	0，内部时钟，Fosc/12
0	1	1，内部时钟，Fosc/2
1	0	2，定时器0溢出,由于定时器0可以工作在1T方式，所以可以达到计一个时钟就溢出，频率反而是最高的，可达到Fosc,通过改变定时器0的溢出率，可以实现可调频率的PWM输出
1	1	3，ECI/P3.4脚的外部时钟输入（最大速率＝Fosc/2）

CCON - PCA 控制寄存器的位分配 （地址：D8H）

位	7	6	5	4	3	2	1	0
符 号	CF	CR	-	-	CCF3	CCF2	CCF1	CCF0

CCON - PCA 控制寄存器的位描述 （地址：D8H）

位	符号	描述
7	CF	PCA计数器阵列溢出标志。计数值翻转时该位由硬件置位。如果CMOD寄存器的ECF位置位，CF标志可用来产生中断。CF位可通过硬件或软件置位，但只可通过软件清零。
6	CR	PCA计数器阵列运行控制位。该位通过软件置位，用来起动PCA计数器阵列计数。该位通过软件清零，用来关闭PCA计数器。
5 - 4	-	保留位，保留为将来使用。
3	CCF3	PCA模块3中断标志。当出现匹配或捕获时该位由硬件置位。该位必须通过软件清零。
2	CCF2	PCA模块2中断标志。当出现匹配或捕获时该位由硬件置位。该位必须通过软件清零。
1	CCF1	PCA模块1中断标志。当出现匹配或捕获时该位由硬件置位。该位必须通过软件清零。
0	CCF0	PCA模块0中断标志。当出现匹配或捕获时该位由硬件置位。该位必须通过软件清零。

CCAPMn - PCA 比较 / 捕获模块寄存器的位分配（CCAPM0 地址：0DAH；CCAPM1 地址：0DBH）

位	7	6	5	4	3	2	1	0
符 号	-	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn

CCAPMn - PCA 比较 / 捕获模块寄存器的位描述 （ n：0,1,2,3 ）

位	符号	描述 n：0,1,2,3
7	-	保留为将来之用。
6	ECOMn	使能比较器。ECOMn＝1时使能比较器功能。
5	CAPPn	正捕获。CAPPn＝1时使能上升沿捕获。
4	CAPNn	负捕获。CAPNn＝1时使能下降沿捕获。
3	MATn	匹配。当MATn＝1时，PCA计数值与模块的比较/捕获寄存器的值的匹配将置位CCON寄存器的中断标志位CCFn。
2	TOGn	翻转。当TOGn＝1时，工作在PCA高速输出模式，PCA计数器的值与模块的比较/捕获寄存器的值的匹配将使CEXn脚翻转。 (CEX0/P3.7,CEX1/P3.5,CEX2/P2.0,CEX3/P2.4)
1	PWMn	脉宽调节模式。当PWMn＝1时，使能CEXn脚用作脉宽调节输出。
0	ECCFn	使能CCFn中断。使能寄存器CCON的比较/捕获标志CCFn，用来产生中断。

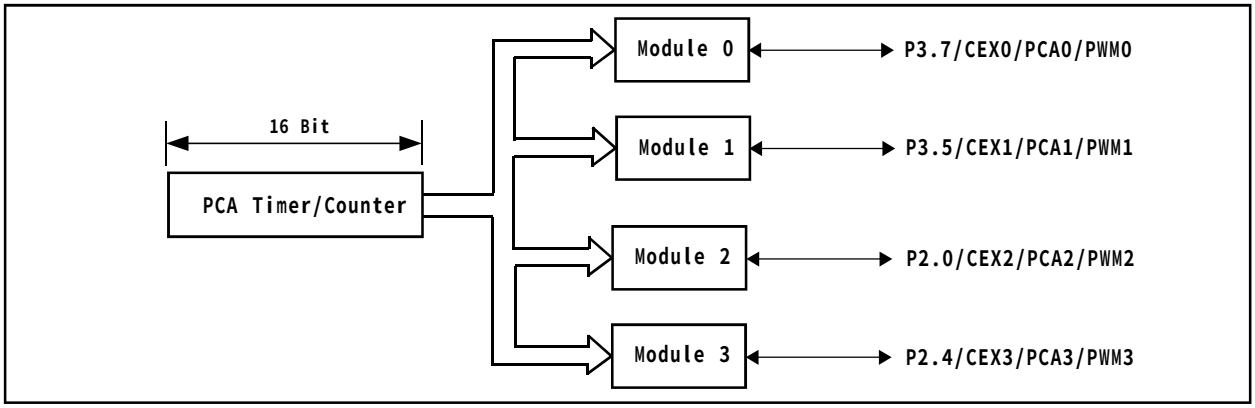
PCA 模块工作模式（CCAPMn 寄存器，n：0,1,2,3）

-	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	模块功能
	0	0	0	0	0	0	0	无此操作
	X	1	0	0	0	0	X	16位捕获模式，由CEXn的上升沿触发
	X	0	1	0	0	0	X	16位捕获模式，由CEXn的下降沿触发
	X	1	1	0	0	0	X	16位捕获模式，由CEXn的跳变触发
	1	0	0	1	0	0	X	16位软件定时器
	1	0	0	1	1	0	X	16位高速输出
	1	0	0	0	0	1	0	8位PWM

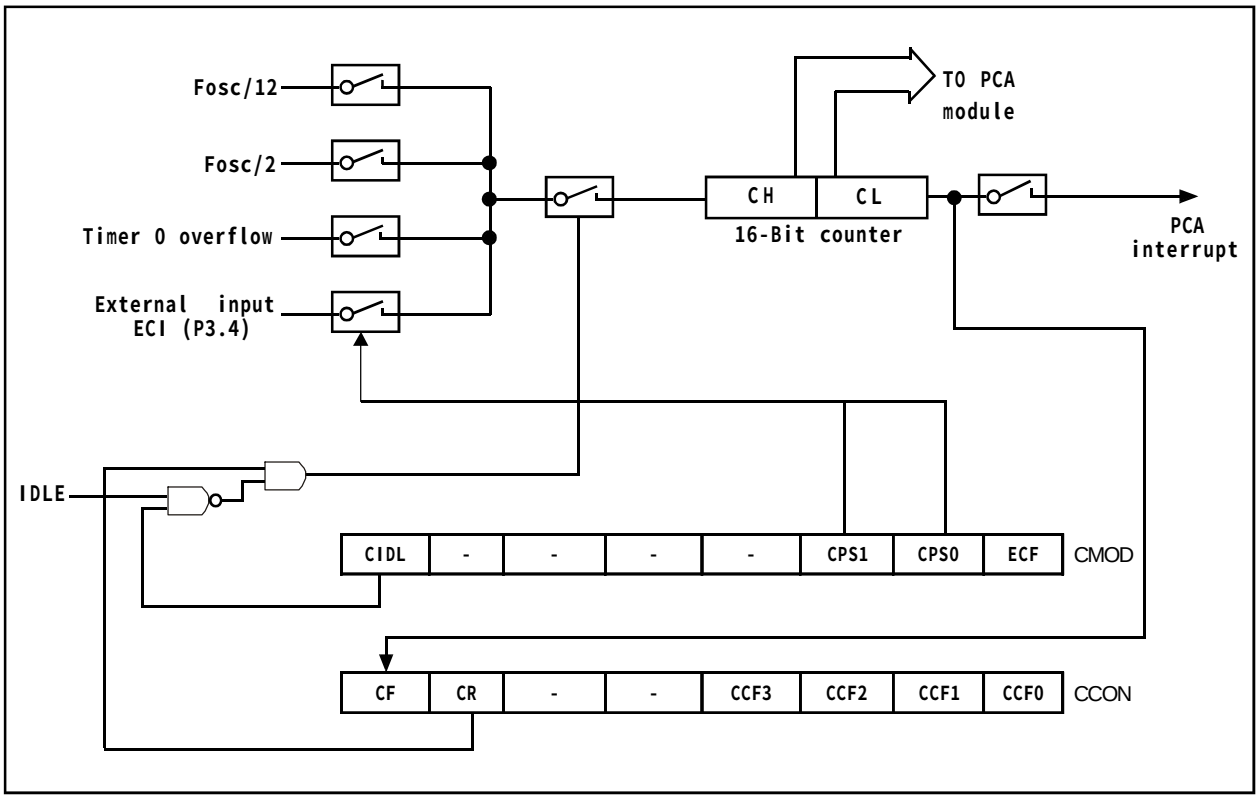
8.2 PCA/PWM 功能介绍

STC12C5410AD 系列单片机有四路可编程计数器阵列（PCA）/PWM，12C2052AD 系列只有两路。

PCA 含有一个特殊的 16 位定时器，有 4 个 16 位的捕获 / 比较模块与之相连。每个模块可编程工作在 4 种模式下：上升 / 下降沿捕获、软件定时器、高速输出或可调制脉冲输出。模块 0 连接到 P3.7（CEX0/PCA0/PWM0），模块 1 连接到 P3.5（CEX1/PCA1/PWM1），模块 2 连接到 P2.0（CEX2/PCA2/PWM2），模块 3 连接到 P2.4（CEX3/PCA3/PWM3）。寄存器 CH 和 CL 的内容是正在自由递增计数的 16 位 PCA 定时器的值。PCA 定时器是 4 个模块的公共时间基准，可通过编程工作在：1/12 振荡频率、1/2 振荡频率、定时器 0 溢出或 ECI 脚的输入（P3.4）。定时器的计数源由 CMOD SFR 的 CPS1 和 CPS0 位来确定（见 CMOD 特殊功能寄存器说明）。



Programmable Counter Array



PCA Timer/Counter

CMOD SFR 还有 2 个位与 PCA 相关。它们分别是：CIDL，空闲模式下允许停止 PCA；ECF，置位时，使能 PCA 中断，当 PCA 定时器溢出将 PCA 计数溢出标志 CF（CCON SFR）置位。

CCON SFR 包含 PCA 的运行控制位（CR）和 PCA 定时器标志（CF）以及各个模块的标志（CCF3/CCF2/CCF1/CCF0)。通过软件置位 CR 位（CCON.6）来运行 PCA。CR 位被清零时 PCA 关闭。当 PCA 计数器溢出时，CF 位（CCON.7）置位,如果 CMOD 寄存器的 ECF 位置位,就产生中断。CF 位只可通过软件清除。CCON 寄存器的位 0～3 是 PCA 各个模块的标志（位 0 对应模块 0，位 1 对应模块 1,位 2 对应模块 2，位 3 对应模块 3)，当发生匹配或比较时由硬件置位。这些标志也只能通过软件清除。所有模块共用一个中断向量。PCA 的中断系统如图所示。

PCA 的每个模块都对应一个特殊功能寄存器。它们分别是：模块 0 对应 CCAPM0，模块 1 对应 CCAPM1，模块 2 对应 CCAPM2，模块 3 对应 CCAPM3.特殊功能寄存器包含了相应模块的工作模式控制位。

当模块发生匹配或比较时，ECCFn 位（CCAPMn.0，n ＝ 0，1，2,3 由工作的模块决定）使能 CCON SFR 的 CCFn 标志来产生中断。

PWM（CCAPMn.1）用来使能脉宽调制模式。

当 PCA 计数值与模块的捕获 / 比较寄存器的值相匹配时，如果 TOG 位（CCAPMn.2）置位，模块的 CEXn 输出将发生翻转。

当 PCA 计数值与模块的捕获 / 比较寄存器的值相匹配时，如果匹配位 MATn（CCAPMn.3）置位，CCON 寄存器的 CCFn 位将被置位。

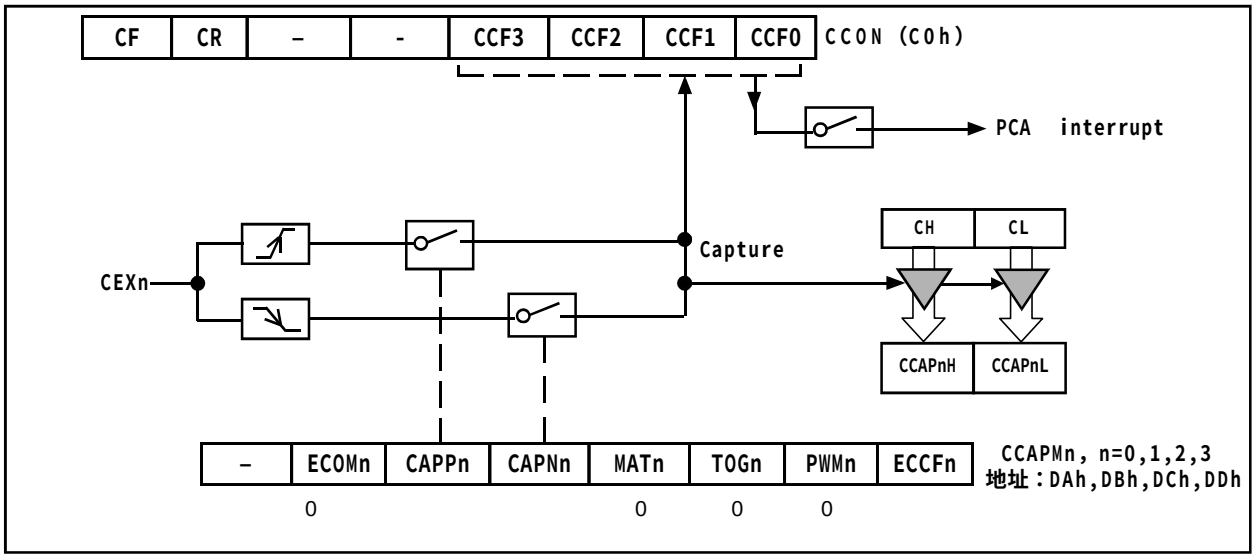
CAPNn（CCAPMn.4）和 CAPPn（CCAPMn.5）用来设置捕获输入的有效沿。CAPNn 位使能下降沿有效，CAPPn 位使能上升沿有效。如果两位都置位，则两种跳变沿都被使能，捕获可在两种跳变沿产生。

通过置位 CCAPMn 寄存器的 ECOMn 位（CCAPMn.6）来使能比较器功能。

每个 PCA 模块还对应另外两个寄存器，CCAPnH 和 CCAPnL。当出现捕获或比较时，它们用来保存 16 位的计数值。当 PCA 模块用在 PWM 模式中时，它们用来控制输出的占空比。

PCA 捕获模式

要使一个 PCA 模块工作在捕获模式（下图），寄存器 CCAPMn 的两位（CAPNn 和 CAPPn）或其中任何一位必须置 1。对模块的外部 CEXn 输入（CEX0/P3.7，CEX1/P3.5,CEX2/P2.0,CEX3/P2.4 口）的跳变进行采样。当采样到有效跳变时，PCA 硬件就将 PCA 计数器阵列寄存器（CH 和 CL）的值装载到模块的捕获寄存器中（CCAPnL 和 CCAPnH）。

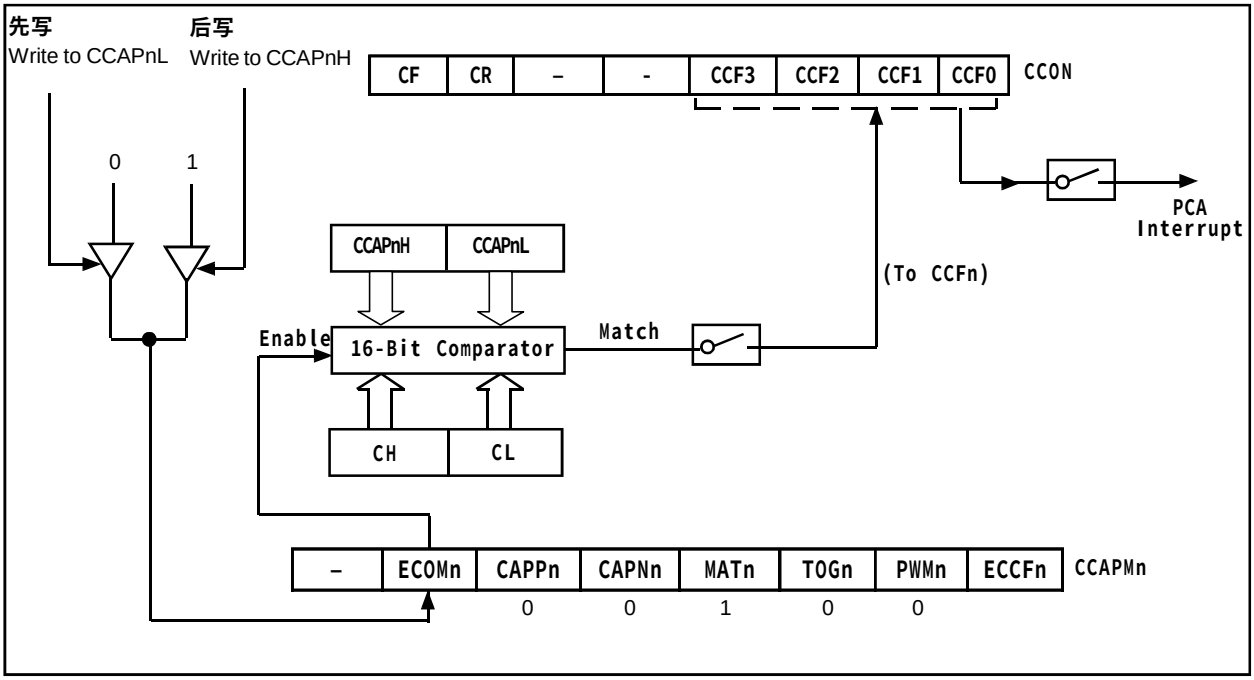


PCA Capture Mode （PCA 捕获模式图）

如果 CCON SFR 的位 CCFn 和 CCAPMn SFR 的位 ECCFn 位被置位，将产生中断。

16 位软件定时器模式

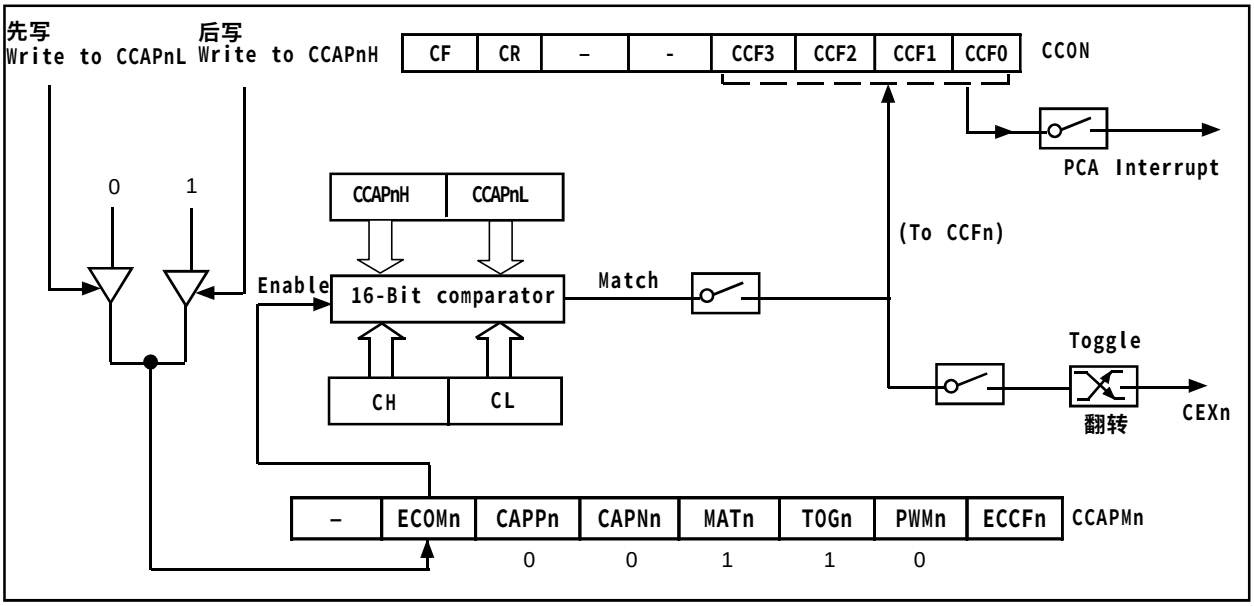
通过置位 CCAPMn 寄存器的 ECOM 和 MAT 位，可使 PCA 模块用作软件定时器（下图）。PCA 定时器的值与模块捕获寄存器的值相比较，当两者相等时，如果位 CCFn（在 CCON SFR 中）和位 ECCFn（在 CCAPMn SFR 中）都置位，将产生中断。



PCA Software Timer Mode/ 软件定时器模式 /PCA 比较模式

高速输出模式

该模式中（下图），当 PCA 计数器的计数值与模块捕获寄存器的值相匹配时，PCA 模块的 CEXn 输出将发生翻转。要激活高速输出模式，模块 CCAPMn SFR 的 TOG,MAT 和 ECOM 位必须都置位。



PCA High-Speed Output Mode / PCA 高速输出模式

在使用 PCA 高速输出模式时的特别应用注意事项：

如果某一 PCA 模块工作在高速脉冲输出模式,要用软件输出改变同一组其它普通 I/O 口的状态，需先做判断 CH 是否等于 CCAPnH，若不等，可自由修改，若相等，再判断 CL>CCAPnL 情况下才允许改变同一组其它普通 I/O 口的状态。如用 P3.7/PCA0/PWM0 做 PCA 高速脉冲输出，同时程序里面又要用软件输出改变 P3.4 口的状态时，就需要做判断。

当某个具有 PCA 高速脉冲输出功能的 I/O 口工作在高速脉冲输出模式时，如果软件对同一组的其它 I/O 口进行操作，如果遇上 PCA 比较器匹配时,该操作有可能会改变此具有 PCA 高速脉冲输出功能的 I/O 口的状态，所以同一组的其它 I/O 口建议不要做输出用，如果做输出用时，要进行判断。

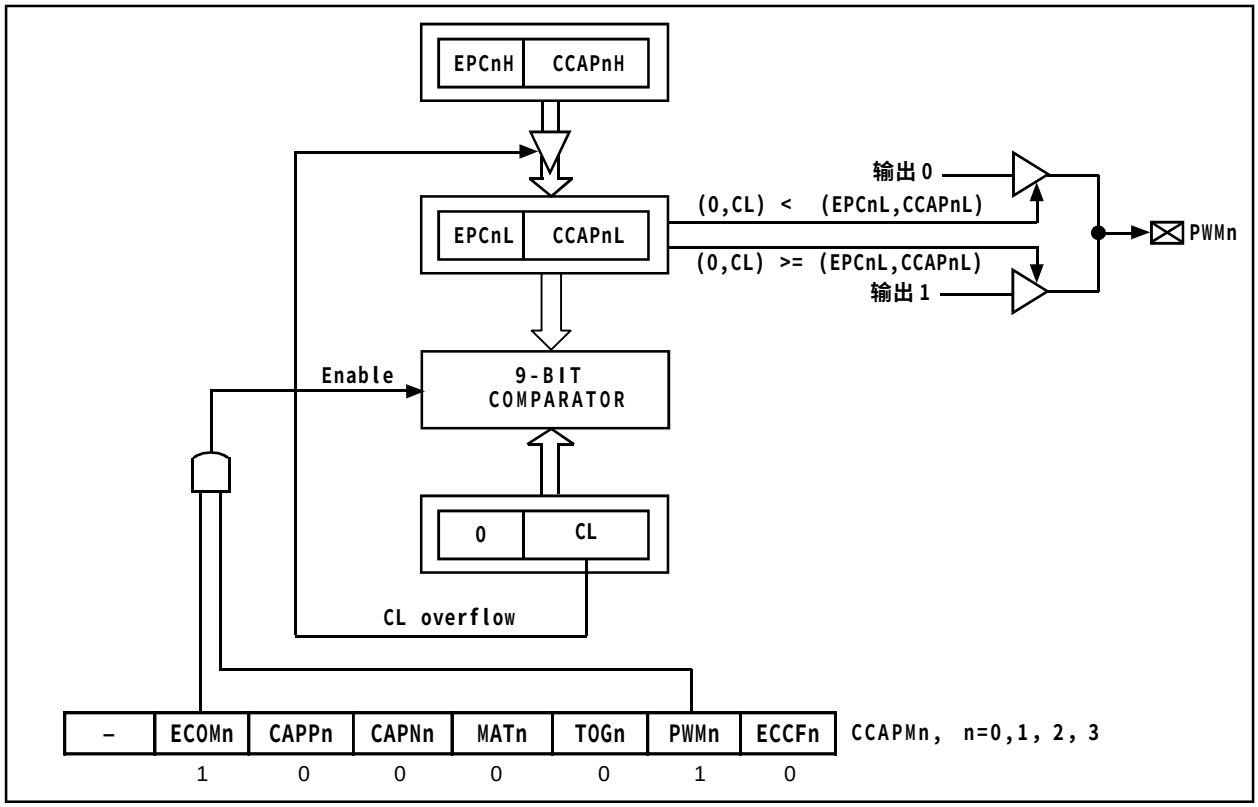
具有PCA高速脉冲输出模式的I/O口 如工作在PCA高速脉冲输出模式	同一组的其它I/O口 建议工作在输入模式，否则软件要注意
PCA0/P3.7	P3.0,P3.1,P3.2,P3.3,P3.4,P3.5
PCA1/P3.5	P3.0,P3.1,P3.2,P3.3,P3.4,P3.7
PCA2/P2.0	P2.1,P2.2,P2.3,P2.4,P2.5,P2.6,P2.7
PCA3/P2.4	P2.0,P2.1,P2.2,P2.3,P2.5,P2.6,P2.7
当以上管脚用在PCA高速脉冲输出模式时，建议同一组的其它I/O口工作在输入模式，如工作在PCA高速输出模式，而同一组的I/O口又必须工作在输出模式，建议如右列所示。	要用软件输出改变同一组其它普通I/O口的状态，需先做判断CH是否等于CCAPnH，若不等，可自由修改，若相等，再判断CL>CCAPnL情况下才允许改变同一组其它普通I/O口的状态。防止在PCA比较器匹配时做输出工作。

其它几种工作模式无问题

更新日期：2008-08-02

脉宽调节模式(PWM)

所有 PCA 模块都可用作 PWM 输出（下图)。输出频率取决于 PCA 定时器的时钟源。



PCA PWM mode / 可调制脉冲宽度输出模式

由于所有模块共用仅有的 PCA 定时器，所有它们的输出频率相同。各个模块的输出占空比是独立变化的，与使用的捕获寄存器 {EPCnL，CCAPnL} 有关。当 CL SFR 的值小于 {EPCnL，CCAPnL} 时，输出为低，当 PCA CL SFR 的值等于或大于 {EPCnL，CCAPnL} 时，输出为高。当 CL 的值由 FF 变为 00 溢出时，{EPCnH，CCAPnH} 的内容装载到 {EPCnL，CCAPnL} 中。这样就可实现无干扰地更新 PWM。要使能 PWM 模式，模块 CCAPMn 寄存器的 PWMn 和 ECOMn 位必须置位。

由于 PWM 是 8 位的，所以：PWM 的频率 = $\frac{\text{PCA时钟输入源频率}}{256}$

PCA 时钟输入源可以从以下 4 种中选择一种：

Fosc / 12， Fosc / 2，定时器 0 的溢出， ECI / P3.4 输入

举例：要求 PWM 输出频率为 38KHz，选 Fosc/2 为 PCA/PWM 时钟输入源，求出 Fosc 的值

由计算公式 38000 = Fosc / 2 / 256 ，得到外部时钟频率 Fosc = 38000 x 256 x 2 =19,456,000

如果要实现可调频率的 PWM 输出,可选择定时器 0 的溢出率或者 ECI 脚的输入作为 PCA/PWM 的时钟输入源

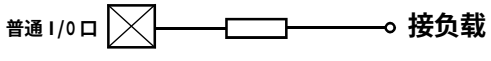
当 EPCnL = 0 及 ECCAPnL = 00H 时,PWM 固定输出高

当 EPCnL = 1 及 CCAPnL = 0FFH 时,PWM 固定输出低

当某个 I/O 口作为 PWM 使用时,该口的状态:

PWM之前口的状态	PWM输出时口的状态
弱上拉 / 准双向口	强推挽输出 / 强上拉输出 ，要加输出限流电阻1K-10K
强推挽输出 / 强上拉输出	强推挽输出 / 强上拉输出 ，要加输出限流电阻1K-10K
仅为输入 / 高阻	PWM无效
开漏	开漏

限流电阻用 10K 到 1K



8.3 用 PCA 功能扩展外部中断的示例程序

```
;/* ---  STC International Limited ---------------------------------- */
;/* ---  宏晶科技     姚永平     2006/1/6     V1.0 ---------------------- */
;/* ---  PCA_5410_ASM_INT ------------------------------------------- */
;/* ---  使用 STC12C2052AD 系列单片机 PCA 功能扩展外部中断的示例程序 --- */
;/* ---  使用 STC12C5410AD 系列单片机 PCA 功能扩展外部中断的示例程序 --- */
;/* ---  STC12C5412AD, STC12C5410AD, STC12C5408AD ------------------- */
;/* ---  STC12C5406AD, STC12C5404AD, STC12C5402AD ------------------- */
;/* ---  STC12C5052AD, STC12C4052AD, STC12C3052AD ------------------- */
;/* ---  STC12C2052AD, STC12C1052AD, STC12C0552AD ------------------- */
;/* ---  Mobile: 13922805190 ---------------------------------------- */
;/* ---  Fax: 0755-82944243 ----------------------------------------- */
;/* ---  Tel: 0755-82948409 ----------------------------------------- */
;/* ---  Web: www.mcu-memory.com ------------------------------------ */
;如果要在程序中使用或在文章中引用该程序,请在程序中或文章中注明使用了宏晶科技的资料及程序
;本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过
;-------------------------------------------------------------------------------
;P3.7(PCA 模块 0) 扩展为下降沿外部中断,
;P3.5(PCA 模块 1) 扩展为上升沿 / 下降沿都可触发的外部中断。
;
;1) 汇编源程序，把汇编程序产生的的程序代码下载到单片机中，上电运行本程序。
;2) 将 P3.7/PCA0 短路到地，这一动作产生一个下降沿，此时本演示程序对 P1.6 取反，
;   P1.6 控制的 LED 灯将会变化一次。
;3) 改变 P3.5/PCA1 的外部高低状态(由高到低 -- 产生下降沿；由低到高 -- 产生上升沿)，
;   本演示程序在 P3.5/PCA1 的下降沿 / 上升沿都产生中断，此时本演示程序对 P1.5 取反，
;   P1.5 控制的 LED 灯状态将会发生变化。
;   所谓 LED 灯状态发生变化是指 LED 由灭变亮或由亮变灭。
;-------------------------------------------------------------------------------
;声明 STC12C2052AD 和 STC12C5410AD 系列 MCU 特殊功能寄存器地址
IPH       EQU   0B7H              ;中断优先级高位寄存器

EPCA_LVD  EQU   IE.6              ;PCA 中断和 LVD(低压检测)中断共享的总中断控制位
CH        EQU   0F9H               ;PCA 计数器高 8 位。
CL        EQU   0E9H               ;PCA 计数器低 8 位。
;------------------------
CCON      EQU   0D8H               ;PCA 控制寄存器。
CCF0      EQU   CCON.0             ;PCA 模块 0 中断标志，由硬件置位，必须由软件清 0。
CCF1      EQU   CCON.1             ;PCA 模块 1 中断标志，由硬件置位，必须由软件清 0。
CCF2      EQU   CCON.2             ;PCA 模块 2 中断标志，由硬件置位，必须由软件清 0。
CCF3      EQU   CCON.3             ;PCA 模块 3 中断标志，由硬件置位，必须由软件清 0。
CCF4      EQU   CCON.4             ;PCA 模块 4 中断标志，由硬件置位，必须由软件清 0。
CCF5      EQU   CCON.5             ;PCA 模块 5 中断标志，由硬件置位，必须由软件清 0。

CR        EQU   CCON.6             ;1:允许 PCA 计数器计数，必须由软件清 0。
CF        EQU   CCON.7             ;PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H)标志,
                                   ;PCA 计数器溢出后由硬件置位,必须由软件清 0。
```

```
;------------------------
CMOD     EQU   0D9H              ;PCA 工作模式寄存器。
;CMOD.7     CIDL: idle 状态时 PCA 计数器是否继续计数，0: 继续计数，1: 停止计数。

;CMOD.2     CPS1: PCA 计数器计数脉冲源选择位 1。
;CMOD.1     CPS0: PCA 计数器计数脉冲源选择位 0。
;           CPS1   CPS0
;            0      0     外部晶体频率 /12。
;            0      1     外部晶体频率 /2。
;            1      0     Timer 0 溢出脉冲，
;                         Timer 0 还可通过 AUXR 寄存器设置成工作在 12T 或 1T 模式。
;            1      1     从 ECI/P3.4 脚输入的外部时钟。

;CMOD.0     ECF: PCA 计数器溢出中断允许位，1-- 允许 CF(CCON.7) 产生中断。
;------------------------

CCAP0H   EQU   0FAH              ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H   EQU   0FBH              ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。
CCAP2H   EQU   0FCH              ;PCA 模块 2 的捕捉 / 比较寄存器高 8 位。
CCAP3H   EQU   0FDH              ;PCA 模块 3 的捕捉 / 比较寄存器高 8 位。
CCAP4H   EQU   0FEH              ;PCA 模块 4 的捕捉 / 比较寄存器高 8 位。
CCAP5H   EQU   0FFH              ;PCA 模块 5 的捕捉 / 比较寄存器高 8 位。

CCAP0L   EQU   0EAH              ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L   EQU   0EBH              ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。
CCAP2L   EQU   0ECH              ;PCA 模块 2 的捕捉 / 比较寄存器低 8 位。
CCAP3L   EQU   0EDH              ;PCA 模块 3 的捕捉 / 比较寄存器低 8 位。
CCAP4L   EQU   0EEH              ;PCA 模块 4 的捕捉 / 比较寄存器低 8 位。
CCAP5L   EQU   0EFH              ;PCA 模块 5 的捕捉 / 比较寄存器低 8 位。

;------------------------
PCA_PWM0 EQU   0F2H              ;PCA 模块 0 PWM 寄存器。
PCA_PWM1 EQU   0F3H              ;PCA 模块 1 PWM 寄存器。
PCA_PWM2 EQU   0F4H              ;PCA 模块 2 PWM 寄存器。
PCA_PWM3 EQU   0F5H              ;PCA 模块 3 PWM 寄存器。
PCA_PWM4 EQU   0F6H              ;PCA 模块 4 PWM 寄存器。
PCA_PWM5 EQU   0F7H              ;PCA 模块 5 PWM 寄存器。

;PCA_PWMn:    7      6      5      4      3      2      1      0
;             -      -      -      -      -      -    EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下，与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下，与 CCAPnL 组成 9 位数。
```

```
;------------------------
CCAPM0   EQU   0DAH              ;PCA 模块 0 的工作模式寄存器。
CCAPM1   EQU   0DBH              ;PCA 模块 1 的工作模式寄存器。
CCAPM2   EQU   0DCH              ;PCA 模块 2 的工作模式寄存器。
CCAPM3   EQU   0DDH              ;PCA 模块 3 的工作模式寄存器。
CCAPM4   EQU   0DEH              ;PCA 模块 4 的工作模式寄存器。
CCAPM5   EQU   0DFH              ;PCA 模块 5 的工作模式寄存器。

;CCAPMn:    7      6      5      4      3      2      1      0
;           -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。

;ECOMn  CAPPn  CAPNn  MATn  TOGn  PWMn  ECCFn
;  0      0      0     0     0     0     0   00H 未启用任何功能。
;  x      1      0     0     0     0     x   21H 16 位 CEXn 上升沿触发捕捉功能。
;  x      0      1     0     0     0     x   11H 16 位 CEXn 下降沿触发捕捉功能。
;  x      1      1     0     0     0     x   31H 16 位 CEXn 边沿(上、下沿)触发捕捉功能。
;  1      0      0     1     0     0     x   49H 16 位软件定时器。
;  1      0      0     1     1     0     x   4DH 16 位高速脉冲输出。
;  1      0      0     0     0     1     0   42H 8 位 PWM。
;-------------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START       EQU P1.7
LED_PCA_INT0        EQU P1.6
LED_PCA_INT1        EQU P1.5
;-------------------------------------------------------------------------------
    ORG   0000H
    LJMP  MAIN
;-------------------------------------------------------------------------------
    ORG   0033H                ;interrupt 6(0,1,2,3,4,5,6)
    LJMP  PCA_Interrupt
;-------------------------------------------------------------------------------
    ORG   0050H
MAIN:
    MOV   SP, #7FH
    CLR   LED_MCU_START        ;点亮 LED_MCU_START LED，表示程序正在运行
    LCALL PCA_Initiate         ;初始化 PCA
WAIT:
    SJMP  WAIT                  ;跳转到本行，无限循环。
```

```
;-------------------------------------------------------------------------------
PCA_Initiate:
    MOV   CMOD, #10000000B ;PCA 在空闲模式下停止 PCA 计数器工作
                           ;PCA 时钟源为 fosc/12
                           ;禁止 PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H 时)中断
    MOV   CCON, #00H       ;CF = 0，清 0 PCA 计数器溢出中断请求标志位
                           ;CR = 0，不允许 PCA 计数器计数
                           ;清 0 PCA 各模块中断请求标志位，如 CCF1，CCF0
    MOV   CL, #00H         ;清 0 PCA 计数器
    MOV   CH, #00H
;------------------------
;设置模块 0
    MOV   CCAPM0, #11H          ;设置 PCA 模块 0 下降沿触发捕捉功能。
;   MOV   CCAPM0, #21H          ;如果送的是 #21h，则 PCA 模块 0 为上升沿触发。
;------------------------
;设置模块 1
    MOV   CCAPM1, #31H          ;设置 PCA 模块 1 上升沿 / 下降沿均可触发的捕捉功能。
;------------------------
    SETB  EPCA_LVD              ;开 PCA 中断和 LVD(低压检测)中断共享的总中断控制位
    SETB  EA                    ;开整个单片机所有中断共享的总中断控制位
    SETB  CR                    ;启动 PCA 计数器(CH,CL)计数
    RET
;-------------------------------------------------------------------------------
PCA_Interrupt:
    PUSH  ACC
    PUSH  PSW

    JNB   CCF0, Not_PCA0_Else_PCA1 ;如果 CCF0 不等于 1 就不是 PCA 模块 0 中断
                                   ;就直接去判是否是 PCA 模块 1 中断
    ;模块 0 中断服务程序
    CPL   LED_PCA_INT0             ;P1.6 LED 变化一次，表示 PCA 模块 0 发生了一次中断
    CLR   CCF0                     ;清 PCA 模块 0 中断标志

Not_PCA0_Else_PCA1:
    JNB   CCF1, PCA_Interrupt_Exit ;如果 CCF1 不等于 1 就不是 PCA 模块 1 中断
                                   ;就立即退出
    ;模块 1 中断服务程序
    CPL   LED_PCA_INT1             ;P1.5 LED 变化一次，表示 PCA 模块 1 发生了一次中断
    CLR   CCF1                     ;清 PCA 模块 1 中断标志

PCA_Interrupt_Exit:
    POP   PSW
    POP   ACC
    RETI
;-------------------------------------------------------------------------------
    END
;-------------------------------------------------------------------------------
```

8.4 用 PCA 功能做定时器的示例程序

```
;/* ---  STC International Limited ---------------------------------- */
;/* ---  宏晶科技     姚永平     2006/1/6     V1.0 ---------------------- */
;/* ---  PCA_5410_ASM_Timer ----------------------------------------- */
;/* ---  使用 STC12C2052AD 系列单片机 PCA 功能做定时器的示例程序 ----- */
;/* ---  使用 STC12C5410AD 系列单片机 PCA 功能做定时器的示例程序 ----- */
;/* ---  STC12C5412AD, STC12C5410AD, STC12C5408AD ------------------ */
;/* ---  STC12C5406AD, STC12C5404AD, STC12C5402AD ------------------ */
;/* ---  STC12C5052AD, STC12C4052AD, STC12C3052AD ------------------ */
;/* ---  STC12C2052AD, STC12C1052AD, STC12C0552AD ------------------ */
;/* ---  Mobile: 13922805190 ---------------------------------------- */
;/* ---  Fax: 0755-82944243 ----------------------------------------- */
;/* ---  Tel: 0755-82948409 ----------------------------------------- */
;/* ---  Web: www.mcu-memory.com ------------------------------------ */
;如果要在程序中使用或在文章中引用该程序,请在程序中或文章中注明使用了宏晶科技的资料及程序
;本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过
;-------------------------------------------------------------------------------
;晶振频率 Fosc = 18.432MHz，在 P1.5 输出脉冲宽度为 1 秒钟的方波
;-------------------------------------------------------------------------------
;声明 STC12C2052AD 和 STC12C5410AD 系列 MCU 特殊功能寄存器地址
IPH       EQU   0B7H              ;中断优先级高位寄存器
EPCA_LVD  EQU   IE.6              ;PCA 中断和 LVD(低压检测)中断共享的总中断控制位
CH        EQU   0F9H               ;PCA 计数器高 8 位。
CL        EQU   0E9H               ;PCA 计数器低 8 位。
;-----------------------
CCON      EQU   0D8H               ;PCA 控制寄存器。
CCF0      EQU   CCON.0             ;PCA 模块 0 中断标志，由硬件置位，必须由软件清 0。
CCF1      EQU   CCON.1             ;PCA 模块 1 中断标志，由硬件置位，必须由软件清 0。
CCF2      EQU   CCON.2             ;PCA 模块 2 中断标志，由硬件置位，必须由软件清 0。
CCF3      EQU   CCON.3             ;PCA 模块 3 中断标志，由硬件置位，必须由软件清 0。
CCF4      EQU   CCON.4             ;PCA 模块 4 中断标志，由硬件置位，必须由软件清 0。
CCF5      EQU   CCON.5             ;PCA 模块 5 中断标志，由硬件置位，必须由软件清 0。
CR        EQU   CCON.6             ;1:允许 PCA 计数器计数，必须由软件清 0。
CF        EQU   CCON.7             ;PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H)标志，
                                   ;PCA 计数器溢出后由硬件置位,必须由软件清 0。
;-----------------------
CMOD      EQU   0D9H               ;PCA 工作模式寄存器。
;CMOD.7     CIDL: idle 状态时 PCA 计数器是否继续计数，0: 继续计数，1: 停止计数。
;CMOD.2     CPS1: PCA 计数器计数脉冲源选择位 1。
;CMOD.1     CPS0: PCA 计数器计数脉冲源选择位 0。
;           CPS1   CPS0
;            0      0    外部晶体频率 /12。
;            0      1     外部晶体频率 /2。
;            1      0     Timer 0 溢出脉冲，
;                         Timer 0 还可通过 AUXR 寄存器设置成工作在 12T 或 1T 模式。
;            1      1     从 ECI/P3.4 脚输入的外部时钟。
;CMOD.0     ECF: PCA 计数器溢出中断允许位，1-- 允许 CF(CCON.7) 产生中断。
```

```
;-----------------------
CCAP0H   EQU   0FAH             ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H   EQU   0FBH             ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。
CCAP2H   EQU   0FCH             ;PCA 模块 2 的捕捉 / 比较寄存器高 8 位。
CCAP3H   EQU   0FDH             ;PCA 模块 3 的捕捉 / 比较寄存器高 8 位。
CCAP4H   EQU   0FEH             ;PCA 模块 4 的捕捉 / 比较寄存器高 8 位。
CCAP5H   EQU   0FFH             ;PCA 模块 5 的捕捉 / 比较寄存器高 8 位。

CCAP0L   EQU   0EAH             ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L   EQU   0EBH             ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。
CCAP2L   EQU   0ECH             ;PCA 模块 2 的捕捉 / 比较寄存器低 8 位。
CCAP3L   EQU   0EDH             ;PCA 模块 3 的捕捉 / 比较寄存器低 8 位。
CCAP4L   EQU   0EEH             ;PCA 模块 4 的捕捉 / 比较寄存器低 8 位。
CCAP5L   EQU   0EFH             ;PCA 模块 5 的捕捉 / 比较寄存器低 8 位。

;-----------------------
PCA_PWM0 EQU   0F2H             ;PCA 模块 0 PWM 寄存器。
PCA_PWM1 EQU   0F3H             ;PCA 模块 1 PWM 寄存器。
PCA_PWM2 EQU   0F4H             ;PCA 模块 2 PWM 寄存器。
PCA_PWM3 EQU   0F5H             ;PCA 模块 3 PWM 寄存器。
PCA_PWM4 EQU   0F6H             ;PCA 模块 4 PWM 寄存器。
PCA_PWM5 EQU   0F7H             ;PCA 模块 5 PWM 寄存器。

;PCA_PWMn:    7      6      5      4      3      2      1      0
;            -      -      -      -      -      -    EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下，与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下，与 CCAPnL 组成 9 位数。

;-----------------------
CCAPM0   EQU   0DAH             ;PCA 模块 0 的工作模式寄存器。
CCAPM1   EQU   0DBH             ;PCA 模块 1 的工作模式寄存器。
CCAPM2   EQU   0DCH             ;PCA 模块 2 的工作模式寄存器。
CCAPM3   EQU   0DDH             ;PCA 模块 3 的工作模式寄存器。
CCAPM4   EQU   0DEH             ;PCA 模块 4 的工作模式寄存器。
CCAPM5   EQU   0DFH             ;PCA 模块 5 的工作模式寄存器。

;CCAPMn:    7      6      5      4      3      2      1      0
;           -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。
```

```
;ECOMn  CAPPn  CAPNn  MATn  TOGn  PWMn  ECCFn
;  0      0      0     0     0     0     0   00H 未启用任何功能。
;  x      1      0     0     0     0     x   21H 16 位 CEXn 上升沿触发捕捉功能。
;  x      0      1     0     0     0     x   11H 16 位 CEXn 下降沿触发捕捉功能。
;  x      1      1     0     0     0     x   31H 16 位 CEXn 边沿(上、下沿)触发捕捉功能。
;  1      0      0     1     0     0     x   49H 16 位软件定时器。
;  1      0      0     1     1     0     x   4DH 16 位高速脉冲输出。
;  1      0      0     0     0     1     0   42H 8 位 PWM。
;------------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START       EQU P1.7
LED_5mS_Flashing    EQU P1.6
LED_1S_Flashing     EQU P1.5
;------------------------------------------------------------------------------
;定义常量
;Channe0_5mS_H, Channe0_5mS_L 的计算方法见 PCA 中断服务程序内的注释
Channe0_5mS_H       EQU 1EH     ;模块 0 5mS 定时常数高位
Channe0_5mS_L       EQU 00H     ;模块 0 5mS 定时常数低位
;------------------------------------------------------------------------------
;定义变量
Counter             EQU 30H     ;声明一个计数器，用来计数中断的次数
;------------------------------------------------------------------------------
    ORG   0000H
    LJMP  MAIN
;------------------------------------------------------------------------------
    ORG   0033H                 ;interrupt 6(0,1,2,3,4,5,6)
    LJMP  PCA_interrupt
;------------------------------------------------------------------------------
    ORG   0050H
MAIN:
    CLR   LED_MCU_START         ;点亮 MCU 开始工作指示灯
    MOV   SP, #7FH
    MOV   Counter, #0           ;清 Counter 计数器

    ACALL PCA_Initiate          ;初始化 PCA
WAIT:
    SJMP  WAIT                    ;跳转到本行，无限循环。
;------------------------------------------------------------------------------
PCA_Initiate:
    MOV   CMOD, #10000000B ;PCA 在空闲模式下停止 PCA 计数器工作
                           ;PCA 时钟源为 fosc/12
                           ;禁止 PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H 时)中断
    MOV   CCON, #00H        ;CF = 0，清 0 PCA 计数器溢出中断请求标志位
                           ;CR = 0，不允许 PCA 计数器计数
                            ;清 0 PCA 各模块中断请求标志位，如 CCF1，CCF0
    MOV   CL, #00H         ;清 0 PCA 计数器
    MOV   CH, #00H
```

```
;-------------------------
    ;Channe0_5mS_H, Channe0_5mS_L 的计算方法见 PCA 中断服务程序内的注释
    MOV   CCAP0L, #Channe0_5mS_L ;给 PCA 模块 0 的 CCAP0L 置初值
    MOV   CCAP0H, #Channe0_5mS_H ;给 PCA 模块 0 的 CCAP0H 置初值
    MOV   CCAPM0, #49H     ;设置 PCA 模块 0 为 16 位软件定时器,ECCF0=1 允许 PCA 模块 0 中断
     ;当[CH, CL]==[CCAP0H, CCAP0L]时，产生中断请求，CCF0=1，请求中断
    SETB  EPCA_LVD              ;开 PCA 中断和 LVD(低压检测)中断共享的总中断控制位
    SETB  EA                    ;开整个单片机所有中断共享的总中断控制位
    SETB  CR                    ;启动 PCA 计数器(CH,CL)计数
    RET
;------------------------------------------------------------------------------
PCA_Interrupt:
    PUSH  ACC
    PUSH  PSW

    CPL   LED_5mS_Flashing  ;本程序 PCA 模块 0 每 5mS 中断一次，每次进中断将该灯状态取反

    ;在本程序中[CH,CL]每 12 个时钟脉冲加 1，当[CH,CL] 增加到等于 [CCAP0H, CCAP0L]时
    ;CCF0=1，产生中断请求。如果每次 PCA 模块 0 中断后，在中断服务程序中给
    ;[CCAP0H, CCAP0L] 增加一个相同的数值，那么下一次中断来临的间隔时间 T 也是相
    ;同的。本程序中这个"相同的数值" 就是 Channe0_5mS_H, Channe0_5mS_L

    ;举例: 时钟频率 Fosc = 18.432MHz, PCA 计数器计数 1E00H 次才是 5mS。
    ;计算 PCA 计数器计数多少次：
    ;     Channe0_5mS_H, Channe0_5mS_L = T/( (1/Fosc)*12 )
    ;                                 = 0.005/ ( (1/18432000)*12 )
    ;                                 = 7680 (10 进制数)
    ;                                 = 1E00H (16 进制数)
    ;     即 Channe0_5mS_H = 1EH, Channe0_5mS_L = 00H
    ;
    ;   Channe0_5mS_H, Channe0_5mS_L：每次给 [CCAP0H,CCAP0L] 增加的数值(步长)

    MOV   A, #Channe0_5mS_L      ;给[CCAP0H, CCAP0L] 增加一个数值
    ADD   A, CCAP0L
    MOV   CCAP0L, A
    MOV   A, #Channe0_5mS_H
    ADDC  A, CCAP0H
    MOV   CCAP0H, A
    CLR   CCF0                  ;清 PCA 模块 0 中断标志

    INC   Counter               ;中断次数计数器 + 1
    MOV   A, Counter
    CLR   C
    SUBB  A, #200                ;检测是否中断了 200 次 (1 秒)
    JC    PCA_Interrupt_Exit    ;有借位，表示 Counter 小于 200，立即跳转退出
```

```
    MOV   Counter, #0           ;已中断了 200 次，清 0 中断次数计数器
    CPL   LED_1S_Flashing       ;在 LED_1S_Flashing 输出脉冲宽度为 1 秒钟的方波

PCA_Interrupt_Exit:
    POP   PSW
    POP   ACC
    RETI
;------------------------------------------------------------------------------
    END
```

8.5 PWM 输出 C 语言示例程序

```
/* ---  STC International Limited ------------------------------------- */
/* ---  宏晶科技    姚永平    2006/1/6    V1.0 ---------------------------- */
/* ---  使用 STC12C2052AD 系列单片机 PWM 输出 C 语言示例程序 ---------------- */
/* ---  使用 STC12C5410AD 系列单片机 PWM 输出 C 语言示例程序 ---------------- */
/* ---  STC12C5412AD, STC12C5410AD, STC12C5408AD ------------------------- */
/* ---  STC12C5406AD, STC12C5404AD, STC12C5402AD ------------------------- */
/* ---  STC12C5052AD, STC12C4052AD, STC12C3052AD ------------------------- */
/* ---  STC12C2052AD, STC12C1052AD, STC12C0552AD ------------------------- */
/* ---  Mobile: 13922805190 ---------------------------------------------- */
/* ---  Fax: 0755-82944243 ----------------------------------------------- */
/* ---  Tel: 0755-82948409 ----------------------------------------------- */
/* ---  Web: www.mcu-memory.com ------------------------------------------ */
/* ---  本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ----------- */
/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 -----*/
/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 -----*/

#include<reg52.h>
sfr  CCON   = 0xD8;
sfr  CMOD   = 0xD9;
sfr  CL     = 0xE9;
sfr  CH     = 0xF9;
sfr  CCAP0L = 0xEA;
sfr  CCAP0H = 0xFA;
sfr  CCAPM0 = 0xDA;
sfr  CCAPM1 = 0xDB;
sbit  CR = 0xDE;
void main(void)
{
    CMOD = 0x02; // Setup PCA timer
    CL = 0x00;
    CH = 0x00;
    CCAP0L = 0xc0;  //Set the initial value same as CCAP0H
    CCAP0H = 0xc0;  //25% Duty Cycle
    CCAPM0 = 0x42;  //0100,0010  Setup PCA module 0 in PWM mode
    CR = 1; //Start PCA Timer.
    while(1){};
 }
```

8.6 PCA/PWM 新增特殊功能寄存器声明

```
;STC12C5410AD 特殊功能寄存器头文件，STC12_PCA_SFR.ASM
;声明 STC12C2052AD 和 STC12C5410AD 系列 MCU 特殊功能寄存器地址
IPH       EQU   0B7H              ;中断优先级高位寄存器
EPCA_LVD  EQU   IE.6              ;PCA 中断和 LVD(低压检测)中断共享的总中断控制位
CH        EQU   0F9H              ;PCA 计数器高 8 位。
CL        EQU   0E9H              ;PCA 计数器低 8 位。
;------------------------
CCON      EQU   0D8H              ;PCA 控制寄存器。
CCF0      EQU   CCON.0            ;PCA 模块 0 中断标志，由硬件置位，必须由软件清 0。
CCF1      EQU   CCON.1            ;PCA 模块 1 中断标志，由硬件置位，必须由软件清 0。
CCF2      EQU   CCON.2            ;PCA 模块 2 中断标志，由硬件置位，必须由软件清 0。
CCF3      EQU   CCON.3            ;PCA 模块 3 中断标志，由硬件置位，必须由软件清 0。
CCF4      EQU   CCON.4            ;PCA 模块 4 中断标志，由硬件置位，必须由软件清 0。
CCF5      EQU   CCON.5            ;PCA 模块 5 中断标志，由硬件置位，必须由软件清 0。
CR        EQU   CCON.6            ;1:允许 PCA 计数器计数，必须由软件清 0。
CF        EQU   CCON.7             ;PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H)标志，
                                  ;PCA 计数器溢出后由硬件置位,必须由软件清 0。
;------------------------
CMOD      EQU   0D9H              ;PCA 工作模式寄存器。
;CMOD.7     CIDL: idle 状态时 PCA 计数器是否继续计数，0: 继续计数，1: 停止计数。

;CMOD.2     CPS1: PCA 计数器计数脉冲源选择位 1。
;CMOD.1     CPS0: PCA 计数器计数脉冲源选择位 0。
;           CPS1   CPS0
;            0      0    外部晶体频率 /12。
;            0      1     外部晶体频率 /2。
;            1      0     Timer 0 溢出脉冲，
;                         Timer 0 还可通过 AUXR 寄存器设置成工作在 12T 或 1T 模式。
;            1      1     从 ECI/P3.4 脚输入的外部时钟。

;CMOD.0     ECF: PCA 计数器溢出中断允许位，1-- 允许 CF(CCON.7) 产生中断。
;------------------------
CCAP0H    EQU   0FAH              ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H    EQU   0FBH              ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。
CCAP2H    EQU   0FCH              ;PCA 模块 2 的捕捉 / 比较寄存器高 8 位。
CCAP3H    EQU   0FDH              ;PCA 模块 3 的捕捉 / 比较寄存器高 8 位。
CCAP4H    EQU   0FEH              ;PCA 模块 4 的捕捉 / 比较寄存器高 8 位。
CCAP5H    EQU   0FFH              ;PCA 模块 5 的捕捉 / 比较寄存器高 8 位。

CCAP0L    EQU   0EAH              ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L    EQU   0EBH              ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。
CCAP2L    EQU   0ECH              ;PCA 模块 2 的捕捉 / 比较寄存器低 8 位。
CCAP3L    EQU   0EDH              ;PCA 模块 3 的捕捉 / 比较寄存器低 8 位。
CCAP4L    EQU   0EEH              ;PCA 模块 4 的捕捉 / 比较寄存器低 8 位。
CCAP5L    EQU   0EFH              ;PCA 模块 5 的捕捉 / 比较寄存器低 8 位。
```

```
;-----------------------
PCA_PWM0 EQU   0F2H             ;PCA 模块 0 PWM 寄存器。
PCA_PWM1 EQU   0F3H             ;PCA 模块 1 PWM 寄存器。
PCA_PWM2 EQU   0F4H             ;PCA 模块 2 PWM 寄存器。
PCA_PWM3 EQU   0F5H             ;PCA 模块 3 PWM 寄存器。
PCA_PWM4 EQU   0F6H             ;PCA 模块 4 PWM 寄存器。
PCA_PWM5 EQU   0F7H             ;PCA 模块 5 PWM 寄存器。

;PCA_PWMn:    7      6      5      4      3      2      1      0
;             -      -      -      -      -      -    EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下，与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下，与 CCAPnL 组成 9 位数。

;-----------------------
CCAPM0   EQU   0DAH             ;PCA 模块 0 的工作模式寄存器。
CCAPM1   EQU   0DBH             ;PCA 模块 1 的工作模式寄存器。
CCAPM2   EQU   0DCH             ;PCA 模块 2 的工作模式寄存器。
CCAPM3   EQU   0DDH             ;PCA 模块 3 的工作模式寄存器。
CCAPM4   EQU   0DEH             ;PCA 模块 4 的工作模式寄存器。
CCAPM5   EQU   0DFH             ;PCA 模块 5 的工作模式寄存器。

;CCAPMn:    7      6      5      4      3      2      1      0
;           -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。
;ECOMn  CAPPn  CAPNn  MATn  TOGn  PWMn  ECCFn
;  0      0      0      0     0     0     0    00H 未启用任何功能。
;  x      1      0      0     0     0     x    21H 16 位 CEXn 上升沿触发捕捉功能。
;  x      0      1      0     0     0     x    11H 16 位 CEXn 下降沿触发捕捉功能。
;  x      1      1      0     0     0     x    31H 16 位 CEXn 边沿(上、下沿)触发捕捉功能。
;  1      0      0      1     0     0     x    49H 16 位软件定时器。
;  1      0      0      1     1     0     x    4DH 16 位高速脉冲输出。
;  1      0      0      0     0     1     0    42H 8 位 PWM。
;-------------------------------------------------------------------------------
```

8.7 PWM 输出汇编语言示例程序

```
;/* ---  STC International Limited ------------------------------------- */
;/* ---  宏晶科技    姚永平    2006/1/6    V1.0 ------------------------- */
;/* ---  使用 STC12C2052AD 系列单片机 PWM 输出汇编语言示例程序 ------------- */
;/* ---  使用 STC12C5410AD 系列单片机 PWM 输出汇编语言示例程序 ------------- */
;/* ---  Mobile: 13922805190 ------------------------------------------- */
;/* ---  Fax: 0755-82944243 -------------------------------------------- */
;/* ---  Tel: 0755-82948409 -------------------------------------------- */
;/* ---  Web: www.mcu-memory.com --------------------------------------- */
;/* ---  本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ---------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 ----- */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 ----- */

;STC12C5410AD 系列单片机 PCA 功能  PWM 示例程序，使用 18.432MHz 晶振。
;-------------------------------------------------------------
#include <..\STC12_PCA_SFR.ASM>     ;定义 PCA 特殊功能寄存器
;-------------------------------------------------------------

;定义常量
;pulse_width_MAX = pulse_width_MIN 时，输出脉冲宽度不变。
pulse_width_MAX   EQU   0F0H        ;PWM 脉宽最大值，占空比 = 93.75%
pulse_width_MIN   EQU   10H         ;PWM 脉宽最小值，占空比 = 6.25%
step              EQU   38H         ;PWM 脉宽变化步长
;-------------------------------------------------------------
;定义变量
pulse_width     EQU  30H
;-------------------------------------------------------------
    ORG   0000H
    AJMP  main
;-------------------------------------------------------------
    ORG   0050H
main:
    MOV   SP, #0E0H

    ACALL PCA_init
main_loop:
    ACALL PWM
    SJMP  main_loop
;-------------------------------------------------------------
```

```
PCA_init:
    MOV   CMOD, #80H;           ;PCA 在空闲模式下停止 PCA 计数器工作
                                ;PCA 时钟模式为 fosc/12
                                ;禁止 PCA 计数器溢出中断
    MOV   CCON, #00H            ;禁止 PCA 计数器工作，清除中断标志、计数器溢出标志
    MOV   CL, #00H              ;清 0 计数器
    MOV   CH, #00H
;------------------------
;设置模块 0 为 8 位 PWM 输出模式，PWM 无需中断支持。脉冲在 P3.7(第 11 脚)输出
    MOV   CCAPM0, #42H          ;*** 示例程序核心语句，---->0100,0010
    MOV   PCA_PWM0, #00H        ;*** 示例程序核心语句
;   MOV   PCA_PWM0, #03H        ;释放本行注释，PWM 输出就一直是 0，无脉冲。

;------------------------
;设置模块 1 为 8 位 PWM 输出模式，PWM 无需中断支持。脉冲在 P3.5(第 9 脚)输出
    MOV   CCAPM1, #42H          ;*** 示例程序核心语句，---->0100,0010
    MOV   PCA_PWM1, #00H        ;*** 示例程序核心语句
;   MOV   PCA_PWM1, #03H        ;释放本行注释，PWM 输出就一直是 0，无脉冲。

    SETB  EPCA_LVD              ;开 PCA 中断
    SETB  EA                    ;开总中断
    SETB  CR                    ;将 PCA 计数器打开
    RET

;-------------------------------------------------------------
PWM:                            ;用示波器进行观察较为理想。

    ;逐渐变亮。
    MOV   A, #pulse_width_MIN   ;为输出脉冲宽度设置初值。
    MOV   pulse_width, A        ;pulse_width 数字越大脉宽越窄，P3.5 的 LED 越亮。
PWM_loop1:
    MOV   A, pulse_width        ;判是否到达最大值。
    CLR   C
    SUBB  A, #pulse_width_MAX
    JNC   PWM_a                 ;到达最大值就转到逐渐变暗。
    MOV   A, pulse_width        ;设置脉冲宽度。数字越大、脉宽越窄、LED 越亮。
    MOV   CCAP0H, A             ;*** 示例程序核心语句
    MOV   CCAP1H, A             ;*** 示例程序核心语句

    CPL   A                     ;用 P1 口的 LED 显示占空比，
    MOV   P1, A                 ;占空比 = ( pulse_width/256 ) * 100%  。

    MOV   A, pulse_width        ;计算下一次输出脉冲宽度数值。
    ADD   A, #step
    MOV   pulse_width, A
    ACALL delay                 ;在一段时间内保持输出脉冲宽度不变。
    SJMP  PWM_loop1
```

```
PWM_a:
     ;逐渐变暗。
    MOV   A, #pulse_width_MAX    ;为输出脉冲宽度设置初值。
    MOV   pulse_width, A         ;pulse_width 数字越大脉宽越窄，P3.5 的 LED 越亮。
PWM_loop2:
    MOV   A, pulse_width         ;判是否到达最小值。
    CLR   C
    SUBB  A, #pulse_width_MIN
    JC    PWM_b                  ;到达最小值就返回。
    JZ    PWM_b                  ;到达最小值就返回。
    MOV   A, pulse_width         ;设置脉冲宽度。数字越大、脉宽越窄、LED 越亮。
    MOV   CCAP0H, A              ;*** 示例程序核心语句
    MOV   CCAP1H, A              ;*** 示例程序核心语句

    CPL   A                      ;用 P1 口的 LED 显示占空比，
    MOV   P1, A              ;占空比 = ( pulse_width/256 ) * 100%  。

    MOV   A, pulse_width     ;计算下一次输出脉冲宽度数值。
    CLR   C
    SUBB  A, #step
    MOV   pulse_width, A
    ACALL delay              ;在一段时间内保持输出脉冲宽度不变。
    SJMP  PWM_loop2
PWM_b:
    RET
;-------------------------------------------------------------

delay:
    CLR   A
    MOV   R1, A
    MOV   R2, A
    MOV   R3, #80H
delay_loop:
    NOP
    NOP
    NOP
    DJNZ  R1, delay_loop
    DJNZ  R2, delay_loop
    DJNZ  R3, delay_loop
    RET
;-------------------------------------------------------------
    END
```

8.8 用 PCA 做高速脉冲输出示例程序

```
;/* ---  STC International Limited ------------------------------------- */
;/* ---  宏晶科技     姚永平     2006/1/6     V1.0 ---------------------------- */
;/* ---  使用 STC12C2052AD 系列单片机 高速脉冲输出功能汇编语言示例程序 ------- */
;/* ---  使用 STC12C5410AD 系列单片机 高速脉冲输出功能汇编语言示例程序 ------- */
;/* ---  Mobile: 13922805190 ------------------------------------------- */
;/* ---  Fax: 0755-82944243 -------------------------------------------- */
;/* ---  Tel: 0755-82948409 -------------------------------------------- */
;/* ---  Web: www.mcu-memory.com --------------------------------------- */
;/* ---  本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ---------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 ---- */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 ---- */
;*******************************************************************************
;                输出 125.0KHz 的脉冲(晶体频率 = 33.000MHz)
;
;示例程序: 使用 功能，在 P3.5(第 9 脚)输出
;          125.0KHz 的方脉冲。
;-------------------------------------------------------------------------------
;    程序中定义的常量 CCAPnL_Value 决定了 PCA 模块 n 输出脉冲的频率 f：
;        f =  Fosc / (4 * CCAPnL_Value )
;           式中 Fosc = 晶体频率
;        CCAPnL_Value  =  Fosc / (4 * f)
;
;    如算出的结果不是整数，则进行取整 CCAPnL_Value =  INT(Fosc / (4 * f) + 0.5)
;        INT() 为取整数运算，直接去掉小数。
;*******************************************************************************
;定义 STC12C5410 系列 MCU 特殊功能寄存器
IPH       EQU   0B7H              ;中断优先级高位寄存器

EPCA_LVD  EQU   IE.6              ;PCA/LVD 中断允许位。
                ;要打开 PCA 中断还要打开相应的 ECF，ECCF0，ECCF1 位
                ;要打开 LVD 中断还要打开相应的 ELVDI 位
CH        EQU   0xF9              ;PCA 计数器高 8 位。
CL        EQU   0xE9              ;PCA 计数器低 8 位。

;------------------------
CCON      EQU   0D8H              ;PCA 控制寄存器。
CCF0      EQU   CCON.0            ;PCA 模块 0 中断标志，由硬件置位，必须由软件清 0。
CCF1      EQU   CCON.1            ;PCA 模块 1 中断标志，由硬件置位，必须由软件清 0。
CR        EQU   CCON.6            ;1:允许 PCA 计数器计数，必须由软件清 0。
CF        EQU   CCON.7            ;PCA 计数器溢出标志,由硬件或软件置位,必须由软件清 0。

;------------------------
```

```
CMOD    EQU   0D9H              ;PCA 工作模式寄存器。
;CMOD.7     CIDL: idle 状态时 PCA 计数器是否继续计数，0: 继续计数，1: 停止计数。

;CMOD.2     CPS1: PCA 计数器脉冲源选择位 1。
;CMOD.1     CPS0: PCA 计数器脉冲源选择位 0。
;           CPS1   CPS0
;            0      0    内部时钟，fosc/12。
;            0      1    内部时钟，fosc/2。
;            1      0    Timer0 溢出。
;            1      1    由 ECI/P3.4 脚输入的外部时钟。
;CMOD.0     ECF: PCA 计数器溢出中断允许位，1-- 允许 CF(CCON.7) 产生中断。
;------------------------
CCAP0H  EQU   0FAH              ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H  EQU   0FBH              ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。
CCAP0L  EQU   0EAH              ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L  EQU   0EBH              ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。
;------------------------
PCA_PWM0 EQU  0F2H              ;PCA 模块 0 PWM 寄存器。
PCA_PWM1 EQU  0F3H              ;PCA 模块 1 PWM 寄存器。

;PCA_PWMn:    7     6     5     4     3     2     1     0
;             -     -     -     -     -     -   EPCnH EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下，与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下，与 CCAPnL 组成 9 位数。
;------------------------
CCAPM0  EQU   0DAH              ;PCA 模块 0 的工作模式寄存器。
CCAPM1  EQU   0DBH              ;PCA 模块 1 的工作模式寄存器。

;CCAPMn:    7     6     5     4     3     2     1     0
;           -   ECOMn CAPPn CAPNn MATn  TOGn  PWMn  ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。

;ECOMn CAPPn CAPNn MATn TOGn PWMn ECCFn
;  0     0     0    0    0    0    0   0x00   未启用任何功能。
;  x     1     0    0    0    0    x   0x21   16 位 CEXn 上升沿触发捕捉功能。
;  x     0     1    0    0    0    x   0x11   16 位 CEXn 下降沿触发捕捉功能。
;  x     1     1    0    0    0    x   0x31   16 位 CEXn 边沿(上、下沿)触发捕捉功能。
;  1     0     0    1    0    0    x   0x49   16 位软件定时器。
;  1     0     0    1    1    0    x   0x4d   16 位高速脉冲输出。
;  1     0     0    0    0    1    0   0x42   8 位 PWM。
```

```
;-------------------------------------------------------------
;定义常量 CCAPnL_Value
;CCAPnL_Value 决定了模块1 输出脉冲的频率 f：
;        f =  Fosc / (4 * CCAPnL_Value )
;    式中 Fosc = 晶体频率
;    或  CCAPnL_Value  =  INT(Fosc / (4 * f) + 0.5)
;        INT() 为取整数运算。
;
;    假定 fosc = 20MHz时，要求 PCA 高速脉冲输出 125KHz 的方波:
;        CCAPnL_Value = INT( 20000000/4/125000 + 0.5)
;                     = INT( 40 + 0.5)
;                     = INT( 40.5 )
;                     = 40
;                     = 28H
;    输出脉冲的频率 f  = 20000000/4/40
;                      = 125000 (125.0KHz)

;CCAPnL_Value  EQU 25H     ;25H = 37, fosc = 18.432MHz时，高速脉冲输出 = 124.540KHz
;CCAPnL_Value  EQU 28H     ;28H = 40, fosc = 20MHz时，高速脉冲输出 = 125KHz
CCAPnL_Value   EQU 42H     ;42H = 66, fosc = 33MHz时，高速脉冲输出 = 125KHz
;-------------------------------------------------------------
    ORG 0000H
    AJMP  main
;-------------------------------------------------------------
    ORG  0033H                  ;interrupt 6
PCA_interrupt:
    PUSH  ACC                   ;4 Clock
    PUSH  PSW                   ;4 Clock

    CLR   CCF1                  ;1 Clock，清 PCA 模块 1 中断标志

    MOV   A, #CCAPnL_Value ;2 Clock
    ADD   A, CCAP1L             ;3 Clock
    MOV   CCAP1L, A             ;3 Clock
    CLR   A                     ;1 Clock
    ADDC  A, CCAP1H             ;3 Clock
    MOV   CCAP1H, A             ;3 Clock

    POP   PSW                   ;3 Clock
    POP   ACC                   ;3 Clock
    RETI                        ;4 Clock

;此中断服务程序共用 34 Clock，进入中断服务程序还要数个 Clock
;-------------------------------------------------------------
```

```
    ORG   0060H
main:
    MOV   SP, #0E0H             ;设置堆栈指针
    ACALL PCA_init              ;调用 PCA 初始化程序

main_loop:
    NOP
    NOP
    NOP
    SJMP  main_loop
;-------------------------------------------------------
PCA_init:                       ;PCA 初始化程序
    MOV   CMOD, #00000010B      ;02H，PCA 计数器在空闲模式下继续工作，CIDL = 0
                  ;PCA 计数器计数脉冲来源为系统时钟源 fosc/2，CPS1，CPS0 = (0,1)
                  ;禁止 PCA 计数器(CH，CL)计数溢出(CH，CL=0000H)中断，ECF = 0
    MOV   CCON, #00H            ;清除 PCA 计数器(CH，CL)计数溢出中断标志，CF = 0
                                ;停止 PCA 计数器(CH，CL)计数，CR = 0
                                ;清除 模块 1 中断标志，CCF1 = 0
                                ;清除 模块 0 中断标志，CCF0 = 0
    MOV   CH, #00H              ;清 0 PCA 计数器高 8 位
    MOV   CL, #00H              ;清 0 PCA 计数器低 8 位
;------------------------
;设置模块 1 为高速脉冲输出模式，脉冲在 P3.5(第 9 脚)输出
    MOV   CCAPM1, #01001101B    ;4DH，设置 PCA 模块 1 为高速脉冲输出模式,允许触发中断
;CCAPMn:    7      6      5      4      3      2      1      0
;           -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;           0      1      0      0      1      1      0      1

    MOV   CCAP1L, #CCAPnL_Value ;给模块 1 置初值，此句不可少
    MOV   CCAP1H, #0 ;给模块 1 置初值，此句不可少

    ;其它中断服务可能会使模块 1 高速脉冲输出的某个周期突然变得很大，因此必须将
    ;PCA 中断的优先级设置为唯一的最高级，其它中断的优先级都要比它低。
    MOV   IPH, #01000000B       ;PCA 中断的优先级设置为唯一的最高级
    MOV   IP, #01000000B

    SETB  EPCA_LVD              ;开 PCA 中断
    SETB  EA                    ;开总中断
    SETB  CR                    ;将 PCA 计数器打开
    RET

;-------------------------------------------------------
    END
;-------------------------------------------------------
```

8.9 利用定时器 0 的溢出作为 PCA 模块的时钟输入源

--- 利用 PCA 模块 0 实现了可调频率的 PWM 输出

--- 利用 PCA 模块 1 重新实现了一个 16 位定时器

```
;/* ---  STC International Limited ------------------------------------- */
;/* ---  宏晶科技     姚永平     2006/1/6     V1.0 ---------------------------- */
;/* ---  使用 STC12C5410AD 系列单片机 定时器 0 的溢出，作为 PCA 模块的时钟输源 - */
;/* ---  实现了可调频率的 PWM 输出,同时利用 PCA 模块再实现了定时器功能 ---------- */
;/* ---  Mobile: 13922805190 ------------------------------------------- */
;/* ---  Fax: 0755-82944243 -------------------------------------------- */
;/* ---  Tel: 0755-82948409 -------------------------------------------- */
;/* ---  Web: www.mcu-memory.com --------------------------------------- */
;/* ---  本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ---------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 ----- */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 ----- */
;-------------------------------------------------------------------
;使用 定时器 0 的溢出，作为 PCA 模块的时钟输入源，利用 PCA 模块的多种功能
;实现了可调频率的 PWM 输出(还可以改变占空比)，同时利用 PCA 模块再实现了定时器功能
;使用 STC12C2052AD 系列单片机 PCA 模块的模块 0 的 PWM 功能 做 PWM 输出的示例程序
;使用 STC12C2052AD 系列单片机 PCA 模块的模块 1 的 16 位软定时器功能做定时器的示例程序
;使用 STC12C5410AD 系列单片机 PCA 模块的模块 0 的 PWM 功能 做 PWM 输出的示例程序
;使用 STC12C5410AD 系列单片机 PCA 模块的模块 1 的 16 位软定时器功能做定时器的示例程序
;晶振频率 Fosc = 18.432MHz，在 P1.5 输出脉冲宽度为 1 秒钟的方波
;-------------------------------------------------------------------------
;声明 STC12C2052AD 和 STC12C5410AD 系列 MCU 特殊功能寄存器地址
IPH       EQU   0B7H              ;中断优先级高位寄存器

EPCA_LVD  EQU   IE.6              ;PCA 中断和 LVD(低压检测)中断共享的总中断控制位
CH        EQU   0F9H              ;PCA 计数器高 8 位。
CL        EQU   0E9H              ;PCA 计数器低 8 位。

;------------------------
CCON      EQU   0D8H              ;PCA 控制寄存器。
CCF0      EQU   CCON.0            ;PCA 模块 0 中断标志，由硬件置位，必须由软件清 0。
CCF1      EQU   CCON.1            ;PCA 模块 1 中断标志，由硬件置位，必须由软件清 0。
CCF2      EQU   CCON.2            ;PCA 模块 2 中断标志，由硬件置位，必须由软件清 0。
CCF3      EQU   CCON.3            ;PCA 模块 3 中断标志，由硬件置位，必须由软件清 0。
CCF4      EQU   CCON.4            ;PCA 模块 4 中断标志，由硬件置位，必须由软件清 0。
CCF5      EQU   CCON.5            ;PCA 模块 5 中断标志，由硬件置位，必须由软件清 0。

CR        EQU   CCON.6            ;1:允许 PCA 计数器计数，必须由软件清 0。
CF        EQU   CCON.7            ;PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H)标志，
                                  ;PCA 计数器溢出后由硬件置位,必须由软件清 0。
```

```
;-----------------------
CMOD    EQU   0D9H              ;PCA 工作模式寄存器。
;CMOD.7     CIDL: idle 状态时 PCA 计数器是否继续计数，0: 继续计数，1: 停止计数。

;CMOD.2     CPS1: PCA 计数器计数脉冲源选择位 1。
;CMOD.1     CPS0: PCA 计数器计数脉冲源选择位 0。
;          CPS1   CPS0
;            0      0    外部晶体频率 /12。
;            0      1    外部晶体频率 /2。
;            1      0    Timer 0 溢出脉冲，
;                        Timer 0 还可通过 AUXR 寄存器设置成工作在 12T 或 1T 模式。
;            1      1    从 ECI/P3.4 脚输入的外部时钟。

;CMOD.0     ECF: PCA 计数器溢出中断允许位，1-- 允许 CF(CCON.7) 产生中断。
;-----------------------

CCAP0H  EQU   0FAH              ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H  EQU   0FBH              ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。
CCAP2H  EQU   0FCH              ;PCA 模块 2 的捕捉 / 比较寄存器高 8 位。
CCAP3H  EQU   0FDH              ;PCA 模块 3 的捕捉 / 比较寄存器高 8 位。
CCAP4H  EQU   0FEH              ;PCA 模块 4 的捕捉 / 比较寄存器高 8 位。
CCAP5H  EQU   0FFH              ;PCA 模块 5 的捕捉 / 比较寄存器高 8 位。

CCAP0L  EQU   0EAH              ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L  EQU   0EBH              ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。
CCAP2L  EQU   0ECH              ;PCA 模块 2 的捕捉 / 比较寄存器低 8 位。
CCAP3L  EQU   0EDH              ;PCA 模块 3 的捕捉 / 比较寄存器低 8 位。
CCAP4L  EQU   0EEH              ;PCA 模块 4 的捕捉 / 比较寄存器低 8 位。
CCAP5L  EQU   0EFH              ;PCA 模块 5 的捕捉 / 比较寄存器低 8 位。

;-----------------------
PCA_PWM0 EQU  0F2H              ;PCA 模块 0 PWM 寄存器。
PCA_PWM1 EQU  0F3H              ;PCA 模块 1 PWM 寄存器。
PCA_PWM2 EQU  0F4H              ;PCA 模块 2 PWM 寄存器。
PCA_PWM3 EQU  0F5H              ;PCA 模块 3 PWM 寄存器。
PCA_PWM4 EQU  0F6H              ;PCA 模块 4 PWM 寄存器。
PCA_PWM5 EQU  0F7H              ;PCA 模块 5 PWM 寄存器。

;PCA_PWMn:    7      6      5      4      3      2      1      0
;             -      -      -      -      -      -    EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下，与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下，与 CCAPnL 组成 9 位数。
;-----------------------
```

```
CCAPM0   EQU   0DAH              ;PCA 模块 0 的工作模式寄存器。
CCAPM1   EQU   0DBH              ;PCA 模块 1 的工作模式寄存器。
CCAPM2   EQU   0DCH              ;PCA 模块 2 的工作模式寄存器。
CCAPM3   EQU   0DDH              ;PCA 模块 3 的工作模式寄存器。
CCAPM4   EQU   0DEH              ;PCA 模块 4 的工作模式寄存器。
CCAPM5   EQU   0DFH              ;PCA 模块 5 的工作模式寄存器。
;CCAPMn:    7      6      5      4      3      2      1      0
;           -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。
;ECOMn  CAPPn  CAPNn  MATn  TOGn  PWMn  ECCFn
;  0      0      0     0     0     0     0   00H 未启用任何功能。
;  x      1      0     0     0     0     x   21H 16 位 CEXn 上升沿触发捕捉功能。
;  x      0      1     0     0     0     x   11H 16 位 CEXn 下降沿触发捕捉功能。
;  x      1      1     0     0     0     x   31H 16 位 CEXn 边沿(上、下沿)触发捕捉功能。
;  1      0      0     1     0     0     x   49H 16 位软件定时器。
;  1      0      0     1     1     0     x   4DH 16 位高速脉冲输出。
;  1      0      0     0     0     1     0   42H 8 位 PWM。
;------------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START       EQU P1.7
LED_5mS_Flashing    EQU P1.6
LED_1S_Flashing     EQU P1.5
;------------------------------------------------------------------------------
;定义常量
;Channe1_5mS_H, Channe1_5mS_L 的计算方法见 PCA 中断服务程序内的注释
;---------------------------------
;用定时器 0 的溢出率作 PCA 计数器(CH,CL)的时钟源时
;Channe1_5mS_H       EQU 03H     ;PCA 模块 1 5mS 定时常数高位，Fosc = 18.432
Channe1_5mS_H        EQU 01H     ;PCA 模块 1 5mS 定时常数高位，Fosc = 18.432
Channe1_5mS_L        EQU 00H     ;PCA 模块 1 5mS 定时常数低位，Fosc = 18.432
;Channe1_5mS_H       EQU 03H     ;PCA 模块 1 5mS 定时常数高位，Fosc = 22.1184
;Channe1_5mS_L       EQU 099H    ;PCA 模块 1 5mS 定时常数低位，Fosc = 22.1184
;-----------------
;内部时钟频率(fosc)/12 作 PCA 计数器(CH,CL)的时钟源
;Channe1_5mS_H       EQU 1EH     ;PCA 模块 1 5mS 定时常数高位
;Channe1_5mS_L       EQU 00H     ;PCA 模块 1 5mS 定时常数低位
;---------------------------------
Timer0_Reload_1      EQU 0F6H    ;Timer0 自动重装数 = -10
Timer0_Reload_2      EQU 0ECH    ;Timer0 自动重装数 = -20
```

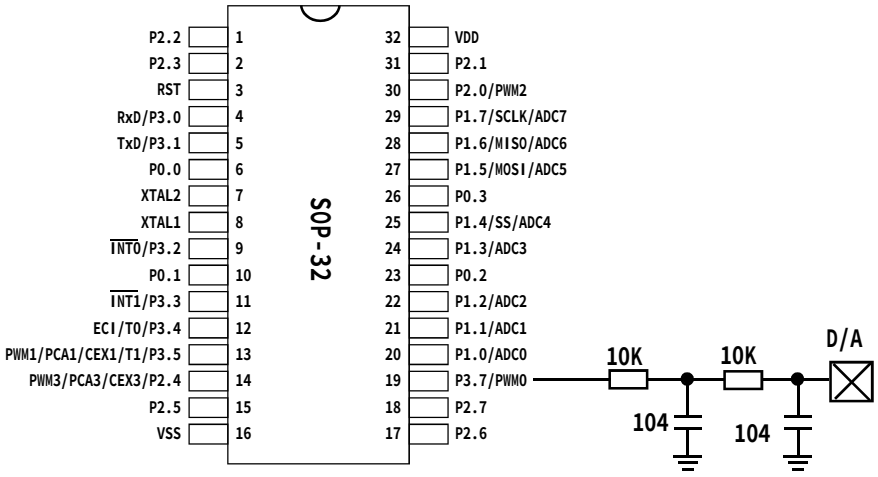
```
;---------------------------------
PWM_PULSE_WIDTH     EQU 0FFH      ;数字越大脉宽越窄(占空比越小)，P3.5 的 LED 越亮。
;-------------------------------------------------------------------------------
;定义变量
Counter             EQU 30H      ;声明一个计数器，用来计数中断的次数
;-------------------------------------------------------------------------------
    ORG   0000H
    LJMP  MAIN
;-------------------------------------------------------------------------------
    ORG   0033H                  ;interrupt 6(0,1,2,3,4,5,6)
    LJMP  PCA_interrupt
;-------------------------------------------------------------------------------
     ORG   0050H
MAIN:
    CLR   LED_MCU_START          ;点亮 MCU 开始工作指示灯
    MOV   SP, #7FH
    MOV   Counter, #0            ;清 Counter 计数器
    ACALL PCA_Initiate           ;初始化 PCA
    ACALL Timer0_Initiate        ;初始化 T0
MAIN_Loop:
;#########  P3.5 的 LED 亮 #################
    MOV   TH0, #Timer0_Reload_1 ;T0 溢出率高
    MOV   TL0, #Timer0_Reload_1

    MOV   A, #PWM_PULSE_WIDTH    ;亮，数字越大 PWM 占空比越小，P3.5 的 LED 越亮。
    MOV   CCAP0H, A
    ACALL delay
;-------------------
;请注意T0溢出率变低后定时器脉冲的 LED 闪烁速度变慢，而 PWM 的 LED 亮度未改变
    MOV   TH0, #Timer0_Reload_2 ;T0 溢出率低
    MOV   TL0, #Timer0_Reload_2
    ACALL delay
;#########  P3.5 的 LED 较亮 #################
    MOV   TH0, #Timer0_Reload_1 ;T0 溢出率高
    MOV   TL0, #Timer0_Reload_1

    MOV   A, #PWM_PULSE_WIDTH
    ACALL RL_A                   ;改变 PWM 占空比
    ACALL RL_A
    MOV   CCAP0H, A              ;较亮，数字越大 PWM 占空比越小，P3.5 的 LED 越亮
    ACALL delay
;-------------------
;请注意T0溢出率变低后定时器脉冲的 LED 闪烁速度变慢，而 PWM 的 LED 亮度未改变
    MOV   TH0, #Timer0_Reload_2 ;T0 溢出率低
    MOV   TL0, #Timer0_Reload_2
    ACALL delay
    MOV   CCAP0H, A              ;暗，数字越大 PWM 占空比越小，P3.5 的 LED 越亮
    ACALL delay
```

```
;#########  P3.5 的 LED 暗 ###################
    MOV   TH0, #Timer0_Reload_1 ;T0 溢出率高
    MOV   TL0, #Timer0_Reload_1

    MOV   A, #PWM_PULSE_WIDTH
    ACALL RL_A                  ;改变 PWM 占空比
    ACALL RL_A
    ACALL RL_A
    ACALL RL_A
;--------------------
;请注意 T0 溢出率变低后定时器脉冲的 LED 闪烁速度变慢，而 PWM 的 LED 亮度未改变
    MOV   TH0, #Timer0_Reload_2 ;T0 溢出率低
    MOV   TL0, #Timer0_Reload_2
     ACALL delay
;##########################################
    SJMP  MAIN_Loop             ;无限循环。
;------------------------------------------------------------------------------
RL_A:
    CLR   C
    RRC   A
    RET
;------------------------------------------------------------------------------
Timer0_Initiate:
;初始化 T0，其溢出脉冲作 PCA 计数器(CH,CL)的时钟源
    MOV   TMOD, #02H            ;设置定时器 0 为自动重装工作模式
    MOV   TH0, #Timer0_Reload_1
    MOV   TL0, #Timer0_Reload_1
    SETB  TR0                   ;启动定时器 0
    RET
;------------------------------------------------------------------------------
PCA_Initiate:
;   MOV   CMOD, #10000000B  ;PCA 在空闲模式下停止 PCA 计数器工作
;                           ;PCA 时钟源为 fosc/12
;                           ;禁止 PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H 时)中断
    MOV   CMOD, #10000100B ;PCA 在空闲模式下停止 PCA 计数器工作
                           ;PCA 时钟源为 定时器 0 (T0) 的溢出率
                           ;禁止 PCA 计数器溢出(CH,CL 由 FFFFH 变为 0000H 时)中断
    MOV   CCON, #00H       ;CF = 0，清 0 PCA 计数器溢出中断请求标志位
                           ;CR = 0，不允许 PCA 计数器计数
                            ;清 0 PCA 各模块中断请求标志位，如 CCF1，CCF0
    MOV   CL, #00H         ;清 0 PCA 计数器
    MOV   CH, #00H
;------------------------
;设置模块 0 为 8 位 PWM 输出模式，PWM 无需中断支持。脉冲在 P3.7(第 11 脚)输出
    MOV   CCAPM0, #42H          ;*** 示例程序核心语句，设置模块 0 为 8 位 PWM 输出模式
    MOV   PCA_PWM0, #00H        ;*** 示例程序核心语句，清 0 PWM 模式下的第 9 位
;   MOV   PCA_PWM0, #03H        ;释放本行注释，PWM 输出就一直是 0，无脉冲。
    MOV   CCAP0H, #PWM_PULSE_WIDTH ;*** 示例程序核心语句
```

```
;-------------------------
;设置 PCA 模块 1
    ;Channe1_5mS_H，Channe1_5mS_L 的计算方法见 PCA 中断服务程序内的注释
    MOV   CCAP1L，#Channe1_5mS_L ;给 PCA 模块 1 的 CCAP1L 置初值
    MOV   CCAP1H，#Channe1_5mS_H ;给 PCA 模块 1 的 CCAP1H 置初值
    MOV   CCAPM1，#49H      ;设置 PCA 模块 1 为 16 位软件定时器,ECCF1=1 允许 PCA 模块 1 中断
     ;当[CH，CL]==[CCAP1H，CCAP1L]时，产生中断请求，CCF1=1，请求中断
    SETB  EPCA_LVD          ;开 PCA 中断和 LVD(低压检测)中断共享的总中断控制位
    SETB  EA                ;开整个单片机所有中断共享的总中断控制位
    SETB  CR                ;启动 PCA 计数器(CH,CL)计数
     RET
;------------------------------------------------------------------------------
PCA_Interrupt:
     PUSH  ACC
     PUSH  PSW

    CPL   LED_5mS_Flashing  ;本程序 PCA 模块 1 每 5mS 中断一次，每次进中断将该灯状态取反

;用定时器 0 的溢出率作 PCA 计数器(CH,CL)的时钟源时，计算 Channe1_5mS_H，Channe1_5mS_L
    ;在本程序中定时器 0 每 12 个时钟脉冲加 1，定时器 0 每加 10 次后产生 1 次溢出，即每
    ;120 个时钟脉冲 PCA 计数器(CH,CL)加 1。当[CH,CL] 增加到等于 [CCAP1H，CCAP1L]时
    ;CCF0=1，PCA 模块 1 产生中断请求。如果每次 PCA 模块 1 中断后，在中断服务程序中给
    ;[CCAP1H，CCAP1L] 增加一个相同的数值，那么下一次中断来临的间隔时间 T 也是相
    ;同的。本程序中这个"相同的数值" 就是 Channe1_5mS_H，Channe1_5mS_L
    ;举例: 时钟频率 Fosc = 18.432MHz，PCA 计数器计数 300H 次等于 5mS。
    ;     Channe1_5mS_H，Channe1_5mS_L = T/( (1/Fosc)*120 )
    ;                                 = 0.005/ ( (1/18432000)*120 )
    ;                                 = 768 (10 进制数)
    ;                                 = 300H (16 进制数)
    ;     即 Channe1_5mS_H = 03H，Channe1_5mS_L = 00H
    ;
    ;   Channe1_5mS_H，Channe1_5mS_L：每次给 [CCAP1H,CCAP1L] 增加的数值(步长)

;内部时钟频率(fosc)/12 作 PCA 计数器(CH,CL)的时钟源,计算 Channe1_5mS_H,Channe1_5mS_L
    ;在本程序中[CH,CL]每 12 个时钟脉冲加 1，当[CH,CL] 增加到等于 [CCAP1H，CCAP1L]时
    ;CCF0=1，PCA 模块 1 产生中断请求。如果每次 PCA 模块 1 中断后，在中断服务程序中给
    ;[CCAP1H，CCAP1L] 增加一个相同的数值，那么下一次中断来临的间隔时间 T 也是相
    ;同的。本程序中这个"相同的数值" 就是 Channe1_5mS_H，Channe1_5mS_L
    ;举例: 时钟频率 Fosc = 18.432MHz，PCA 计数器计数 1E00H 次才是 5mS。
    ;     Channe1_5mS_H，Channe1_5mS_L = T/( (1/Fosc)*12 )
    ;                                 = 0.005/ ( (1/18432000)*12 )
    ;                                 = 7680 (10 进制数)
    ;                                 = 1E00H (16 进制数)
    ;     即 Channe1_5mS_H = 1EH，Channe1_5mS_L = 00H
     ;
    ;   Channe1_5mS_H，Channe1_5mS_L：每次给 [CCAP1H,CCAP1L] 增加的数值(步长)
```

```
    MOV   A, #Channe1_5mS_L     ;给[CCAP1H，CCAP1L] 增加一个数值
    ADD   A, CCAP1L
    MOV   CCAP1L, A
    MOV   A, #Channe1_5mS_H
    ADDC  A, CCAP1H
    MOV   CCAP1H, A
    CLR   CCF1                  ;清 PCA 模块 1 中断标志

    INC   Counter               ;中断次数计数器 + 1
    MOV   A, Counter
    CLR   C
     SUBB  A, #100              ;检测是否中断了 100 次（0.5 秒）
    JC    PCA_Interrupt_Exit    ;有借位，表示 Counter 小于 100，立即跳转退出

    MOV   Counter, #0           ;已中断了 100 次，清 0 中断次数计数器
    CPL   LED_1S_Flashing       ;在 LED_1S_Flashing 输出脉冲宽度为 0.5 秒钟的方波

PCA_Interrupt_Exit:
    POP   PSW
    POP   ACC
    RETI
;-------------------------------------------------------------------------------
delay:
    CLR   A
    MOV   R1, A
    MOV   R2, A
    MOV   R3, #80H
delay_loop:
    NOP
    NOP
    NOP
    DJNZ  R1, delay_loop
    DJNZ  R2, delay_loop
    DJNZ  R3, delay_loop
    RET
;-------------------------------------------------------------------------------
    END
```

8.10 利用 PWM 实现 D/A 功能的典型应用电路图



第九章 STC12 系列单片机的电源管理及掉电模式

9.1 电源管理寄存器 PCON 的应用

- - - - 上电标志位，低压检测标志位

- - - - 掉电模式，空闲模式

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000

LVDF：内部电源低电压检测标志位

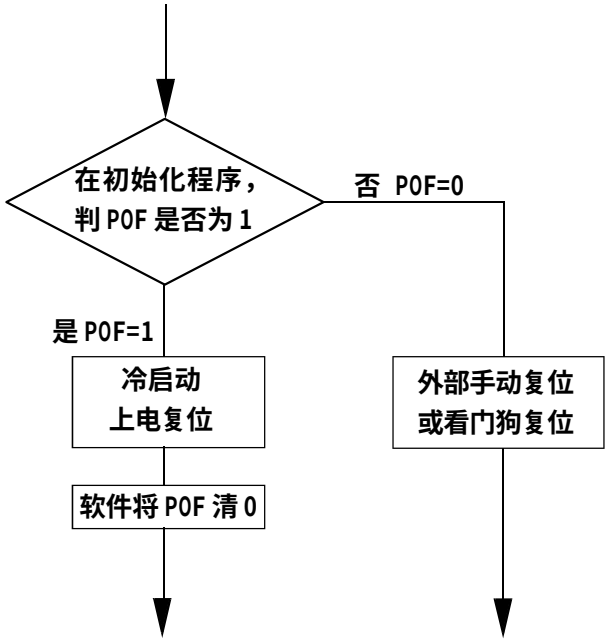
当单片机工作电压低于检测门槛电压时，该位置“1”，该位只能通过软件清零， 如果允许内部低电压检测中断，则该位为低电压中断请求标志位。

5V 单片机在 Vcc<3.7V 时，LVDF = 1，Vcc 〉3.7V 时，LVDF 的值不变，该位只能通过软件清零；

3V 单片机在 Vcc<2.4V 时，LVDF = 1，Vcc 〉2.4V 时，LVDF 的值不变，该位只能通过软件清零；

POF：上电复位标志位，单片机停电后再上电（冷启动），POF 上电标志位为 1，只能由软件清 0。

实际应用：要判断是上电复位（冷启动），还是外部复位脚输入复位信号产生的复位，还是内部看门狗复位，可通过如下方法来判断：



PD：将其置 1 时，进入 Power Down 模式，可由外部中断低电平触发或下降沿触发中断模式唤醒。

进入掉电模式时，外部时钟停振，CPU、定时器、串行口全部停止工作，只有外部中断继续工作。

IDL：将其置 1，进入 IDLE 模式（空闲），除 CPU 不工作外，其余仍继续工作，可由任何一个中断唤醒。

GF1,GF0： 两个通用工作标志位,用户可以任意使用。

SMOD： 波特率倍速位，置 1，串口通讯波特率快一倍

9.2 利用外部中断实现单片机从掉电模式唤醒(C 语言)

```
/* ---  STC International Limited ------------------------------------------ */
/* ---  宏晶科技    姚永平    2006/8/2    V1.0 ------------------------------  */
/* ---  STC8912C5410 系列单片机,掉电模式唤醒测试程序(从外部中断 0 唤醒)-------------*/
/* ---  Mobile: 13922805190 ------------------------------------------------ */
/* ---  Fax: 0755-82944243 ------------------------------------------------- */
/* ---  Tel: 0755-82948409 ------------------------------------------------- */
/* ---  Web: www.mcu-memory.com -------------------------------------------- */
/* ---  本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ----------------- */
/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 ----------- */
/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序----------- */

#include<reg52.h>
#include<intrins.h>

sbit  Begin_Led = P1^2;           // 系统开始工作指示灯
unsigned char Is_Power_Down = 0; // 进入 Power Down 之前，将其置为 1，以供判断
sbit  Is_Power_Down_Led_INT0     = P1^7; // 掉电唤醒指示灯，在外部中断 0 中
sbit  Not_Power_Down_Led_INT0    = P1^6; // 不是掉电唤醒指示灯，在外部中断 0 中
sbit  Is_Power_Down_Led_INT1     = P1^5; // 掉电唤醒指示灯，在外部中断 1 中
sbit  Not_Power_Down_Led_INT1    = P1^4; // 不是掉电唤醒指示灯，在外部中断 1 中
sbit  Power_Down_Wakeup_Pin_INT0 = P3^2; // 掉电唤醒管脚，外部中断 0
sbit  Power_Down_Wakeup_Pin_INT1 = P3^3; // 掉电唤醒管脚，外部中断 1
sbit  Normal_Work_Flashing_Led   = P1^3; // 系统处于正常工作状态指示灯


void Normal_Work_Flashing(void);
void INT_System_init(void);
void INT0_Routine(void);
void INT1_Routine(void);

void main(void)
{
    unsigned  char  j  =  0;
    unsigned  char  wakeup_counter  =  0;  // 中断唤醒次数变量初始为 0
    Begin_Led  =  0;       // 系统开始工作指示灯
    INT_System_init();     // 中断系统初始化
    while(1)
    {
        P2  = ~wakeup_counter;    // 中断唤醒次数显示,先将 wakeup_counter 取反
        wakeup_counter++;         // 中断唤醒次数显示
        for(j=0;j<2;j++)
        {
              Normal_Work_Flashing();    // 系统正常工作指示灯
        }
         Is_Power_Down  =  1; // 进入 Power Down 之前，将其置为 1，以供判断
         PCON  =  0x02; // 执行完此句，单片机进入 Power Down 模式，外部时钟停止振荡
```

```
        _nop_();
        //STC12 系列掉电模式，外部中断唤醒后，首先执行上句，然后才会进入中断服务程序
        _nop_();
        //STC89 系列掉电模式，外部中断唤醒后，首先执行上句，然后才会进入中断服务程序
        _nop_();  // 建议多加几个空操作指令 NOP
        _nop_();  // 建议多加几个空操作指令 NOP
    }
}

void INT_System_init(void)
{
    IT0   =   0;   /* 外部中断 0，低电平触发中断 */
//  IT0   =   1;   /* 外部中断 0，下降沿触发中断 */
    EX0   =   1;   /* 允许外部中断 0 中断 */
    IT1   =   0;   /* 外部中断 1，低电平触发中断 */
//  IT1   =   1;   /* 外部中断 1，下降沿触发中断 */
    EX1   =   1;   /* 允许外部中断 1 中断 */
    EA    =   1;   /* 开总中断控制位 */
}

void INT0_Routine(void) interrupt 0
{
    if(Is_Power_Down)
    {   //Is_Power_Down ==1,掉电唤醒,在外部中断 0 中
        Is_Power_Down  =   0;
        Is_Power_Down_Led_INT0  =  0;  // 点亮外部中断 0 掉电唤醒指示灯
        while(Power_Down_Wakeup_Pin_INT0==0)
        {
            /* 等待变高 */
        }
        Is_Power_Down_Led_INT0  =  1;  // 关闭外部中断 0 掉电唤醒指示灯
    }
    else
    {
        Not_Power_Down_Led_INT0 =  0;  // 点亮外部中断 0 正常工作中断指示灯
        while(Power_Down_Wakeup_Pin_INT0==0)
        {
            /* 等待变高 */
        }
        Not_Power_Down_Led_INT0 =  1;  // 关闭外部中断 0 正常工作中断指示灯
    }
}

void INT1_Routine(void) interrupt 2
{
    if(Is_Power_Down)
    {   //Is_Power_Down ==1,掉电唤醒,在外部中断 1 中
        Is_Power_Down   =   0;
```

```
        Is_Power_Down_Led_INT1 =  0;  //顶亮外部中断1掉电唤醒指示灯
        while(Power_Down_Wakeup_Pin_INT1==0)
        {
            /* 等待变高 */
        }
        Is_Power_Down_Led_INT1  =  1;  //关闭外部中断1掉电唤醒指示灯
    }
    else
    {
        Not_Power_Down_Led_INT1  =  0;  //顶亮外部中断1正常工作中断指示灯
        while(Power_Down_Wakeup_Pin_INT1==0)
        {
            /* 等待变高 */
        }
        Not_Power_Down_Led_INT1  =  1;  //关闭外部中断1正常工作中断指示灯
    }
}
void delay(void)
{
    unsigned int    j   =   0x00;
    unsigned int    k   =   0x00;
    for(k=0;k<2;++k)
    {
        for(j=0;j<=30000;++j)
        {
            _nop_();
            _nop_();
            _nop_();
            _nop_();
            _nop_();
            _nop_();
            _nop_();
            _nop_();
        }
    }
}
void Normal_Work_Flashing(void)
{
    Normal_Work_Flashing_Led    =   0;
    delay();
    Normal_Work_Flashing_Led    =   1;
    delay();
}
```

9.3 通过外部中断从掉电模式唤醒

```
;*************************************************************************
;Wake Up Idle and Wake Up Power Down
;*************************************************************************
    ORG   0000H
    AJMP  MAIN

    ORG   0003H
int0_interrupt:
    CLR   P1.7          ;点亮 P1.7 LED 表示已响应 int0 中断
    ACALL delay         ;延时是为了便于观察，实际应用不需延时
    CLR   EA            ;关闭中断，简化实验．实际应用不需关闭中断
    RETI

    ORG   0013H
int1_interrupt:
    CLR   P1.6          ;点亮 P1.6 LED 表示已响应 int1 中断
    ACALL delay         ;延时是为了便于观察，实际应用不需延时
    CLR   EA            ;关闭中断，简化实验．实际应用不需关闭中断
    RETI

    ORG   0100H
delay:
    CLR   A
    MOV   R0, A
    MOV   R1, A
    MOV   R2, #02
delay_loop:
    DJNZ  R0, delay_loop
    DJNZ  R1, delay_loop
    DJNZ  R2, delay_loop
    RET

main:
    MOV   R3, #0        ;P1 LED 递增方式变化，表示程序开始运行
main_loop:
    MOV   A, R3
    CPL   A
    MOV   P1, A
    ACALL delay
```

```
    INC   R3
    MOV   A, R3
    SUBB  A, #18H
    JC    main_loop

    MOV   P1, #0FFH      ;熄灭全部灯表示进入 Power Down 状态

    CLR   IT0            ;设置低电平激活外部中断
;   SETB  IT0

    SETB  EX0            ;允许外部中断 0

    CLR   IT1            ;设置低电平激活外部中断
;   SETB  IT1
    SETB  EX1            ;允许外部中断 1
;   SETB  ET0  ;如果是 STC12C2052AD 系列 A 版本,
               ;要由外部中断 1 唤醒,“ ET0=1 ” 是必须的,硬件就这样做的,C 版本就不需要
               ;外部中断 0 就无此必要,建议 Powerdown 用外部中断 0 唤醒

    SETB  EA             ;开中断, 若不开中断就不能唤醒 Power Down

;下条语句将使 MCU 进入 idle 状态或 Power Down 状态
;低电平激活外部中断可以将 MCU 从 Power Down 状态中唤醒
;其方法为:将外部中断脚拉低

    MOV   PCON, #00000010B     ;令 PD=1, 进入 Power Down 状态, PD = PCON.1

    ;MOV   PCON, #00000001B    ;删除本语句前的";", 同时将前 1 条语句前加上注释符号 ";",
                               ;令 IDL=1, 可进入 idle 状态, IDL = PCON.0

    MOV   P1, #0DFH      ;1101,1111  请注意:
                         ; 1.外部中断使 MCU 退出 Power Down 状态,执行本条指令后
                         ;响应中断, 表现为 P1.5 与 P1.7 的 LED 同时亮(INT0 唤醒)
                         ; 2.外部中断使 MCU 退出 idle 状态,先响应中断然后再执行本
                         ;条指令, 表现为 P1.7 的 LED 先亮(INT0 唤醒)P1.5 的 LED 后亮
                         ; 3.实际使用掉电模式时,本语句应用 NOP 代替
    NOP       ;实际使用掉电模式时,应在 MOV   PCON, #00000010B 语句后面多加几个 NOP
    NOP       ;实际使用掉电模式时,应在 MOV   PCON, #00000010B 语句后面多加几个 NOP
    NOP       ;实际使用掉电模式时,应在 MOV   PCON, #00000010B 语句后面多加几个 NOP
WAIT1：
    SJMP  WAIT1          ;跳转到本语句, 停机
    END
;A 版本和 B 版本建议不要用 IDLE 模式,现 C 版本可以正常使用
```

第十章 STC12C5410AD 系列单片机电气特性
STC12C2052AD 系列单片机电气特性

ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings

Parameter	Symbol	MIN	MAX	UNIT
Storage temperature	T_{ST}	-55	+125	℃
Operating Temperature(I)	T_A	-40	+85	℃
Operating Temperature(C)	T_A	0	+70	℃
DC Power Supply(5V MCU)	V_{DD} - V_{SS}	-0.3	+5.5	V
DC Power Supply(3V MCU)	V_{DD} - V_{SS}	-0.3	+3.8	V
Voltage on any Pin		-0.5	+5.5	V

DC Specification(5V MCU)

Symbol	Parameter	Specification				Test Condition
		Min.	Typ.	Max.	Unit	
V_{DD}	Operating Voltage	3.5	5.0	5.5	V	
I_{PWDN}	Power Down Current		<0.1		uA	5V
I_{IDLE}	Idle Current		3.0		mA	5V
I_{CC}	Operating Current		4 mA	20	mA	5V
V_{IL1}	Input low voltage (P0,P1,P2,P3)			0.8	V	5V
V_{IL2}	Input low voltage (RESET,XTAL1)			1.2	V	5V
V_{IH1}	Input High voltage (P0,P1,P2,P3)	2.2			V	5V
V_{IH2}	Input High voltage (RESET,XTAL1)	2.2			V	5V
I_{OL1}	Sinking Current for Output Low (P0,P1,P2,P3)		20		mA	5V
I_{OH1}	Sourcing Current for Output High (P0,P1,P2,P3)	150	230		uA	5V
I_{OH2}	Sourcing Current for Output High (P0,P1,P2,P3) (Push-Pull)		20		mA	5V
I_{IL}	Logic 0 input current (P0,P1,P2,P3)		18	50	uA	V_{PIN}=0V
I_{TL}	Logic 1 to 0 transition current (P0,P1,P2,P3)		270	600	uA	V_{PIN}=2V

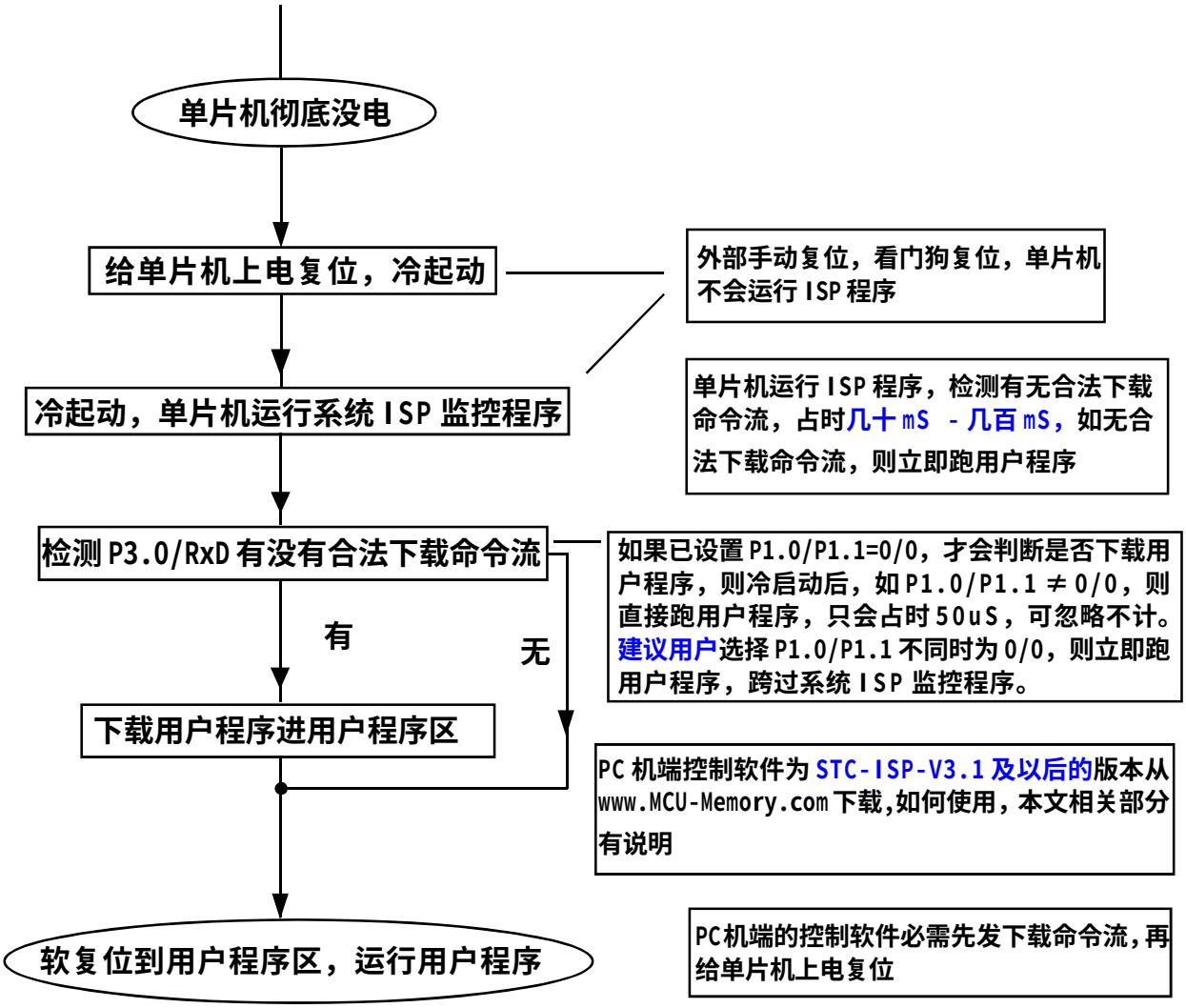
DC Specification(3.3V MCU)

Symbol	Parameter	Specification				Test Condition
		Min.	Typ.	Max.	Unit	
V_{DD}	Operating Voltage	2.2	3.3	3.8	V	
I_{PWDN}	Power Down Current		<0.1		uA	3.3V
I_{IDLE}	Idle Current		2.0		mA	3.3V
I_{CC}	Operating Current		4 mA	10	mA	3.3V
V_{IL1}	Input low voltage (P0,P1,P2,P3)			0.8	V	3.3V
V_{IL2}	Input low voltage (RESET,XTAL1)			1.2	V	3.3V
V_{IH1}	Input High voltage (P0,P1,P2,P3)	2.2			V	3.3V
V_{IH2}	Input High voltage (RESET,XTAL1)	2.2			V	3.3V
I_{OL1}	Sinking Current for Output Low (P0,P1,P2,P3)		20		mA	3.3V
I_{OH1}	Sourcing Current for Output High (P0,P1,P2,P3)	40	70		uA	3.3V
I_{OH2}	Sourcing Current for Output High (P0,P1,P2,P3) (Push-Pull)		20		mA	3.3V
I_{IL}	Logic 0 input current (P0,P1,P2,P3)		8	50	uA	V_{PIN}=0V
I_{TL}	Logic 1 to 0 transition current (P0,P1,P2,P3)		110	600	uA	V_{PIN}=2V

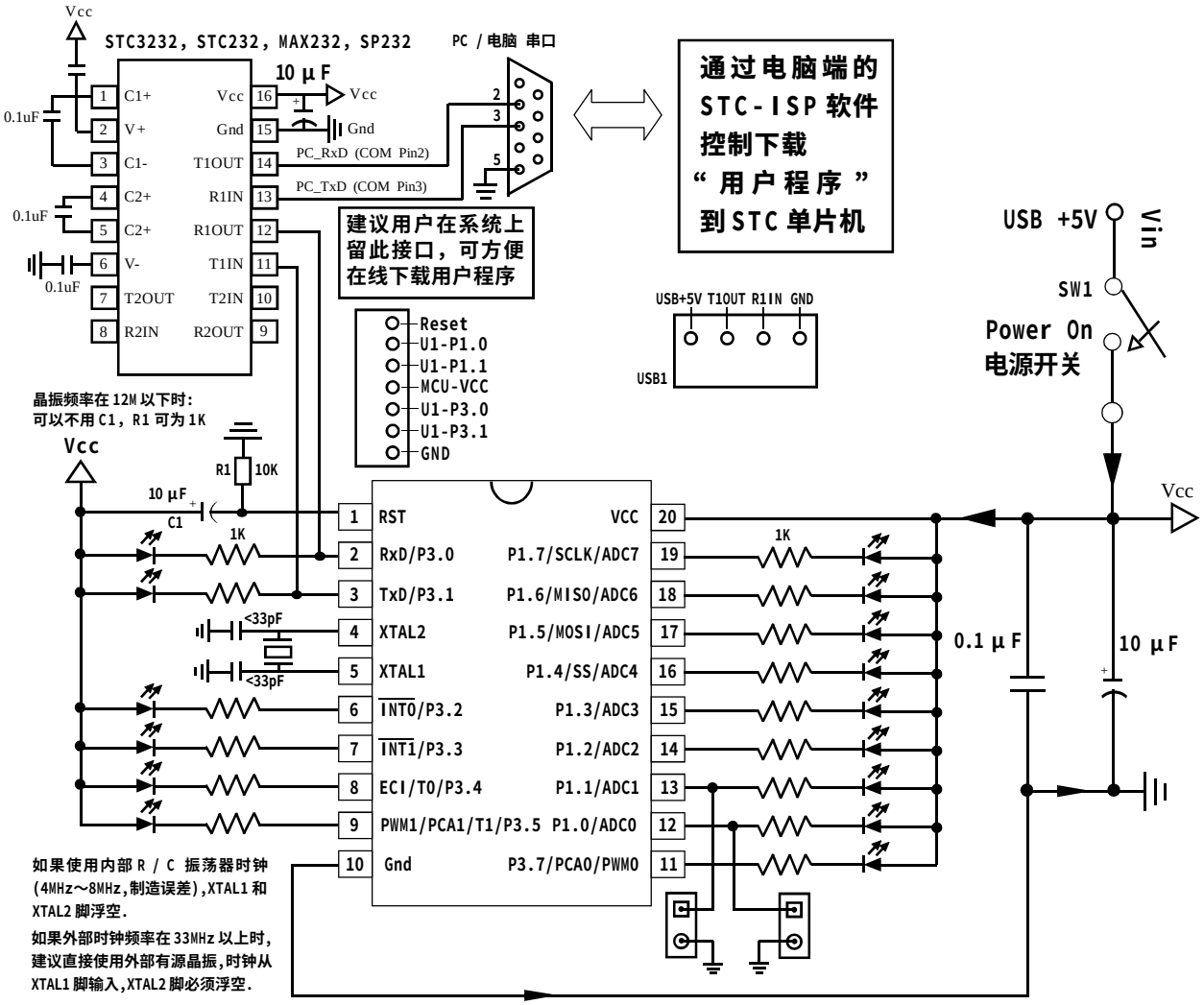
第十一章 STC12系列单片机开发/编程工具说明

11.1 在系统可编程（ISP）原理，官方演示工具使用说明

11.1.1 在系统可编程（ISP）原理使用说明



11.1.2 在系统可编程（ISP）典型应用线路图



STC12C5410AD 及 STC12C2052AD 系列单片机具有在系统可编程（ISP）特性，ISP 的好处是：省去购买通用编程器，单片机在用户系统上即可下载/烧录用户程序，而无须将单片机从已生产好的产品上拆下，再用通用编程器将程序代码烧录进单片机内部。有些程序尚未定型的产品可以一边生产，一边完善，加快了产品进入市场的速度，减小了新产品由于软件缺陷带来的风险。由于可以在用户的目标系统上将程序直接下载进单片机看运行结果对错，故无须仿真器。

STC12 系列单片机内部固化有 ISP 系统引导固件，配合 PC 端的控制程序即可将用户的程序代码下载进单片机内部，故无须编程器(速度比通用编程器快，几秒一片)。

如何获得及使用 STC 提供的 ISP 下载工具（STC-ISP.exe 软件）：

(1). 获得 STC 提供的 ISP 下载工具（软件）

登陆 www.MCU-Memory.com 网站，从 STC 半导体专栏下载 PC（电脑）端的 ISP 程序，然后将其自解压，再安装即可（执行 setup.exe),注意随时更新软件。

(2). 使用 STC-ISP 下载工具（软件),请随时更新，目前已到 Ver3.1 版本以上，

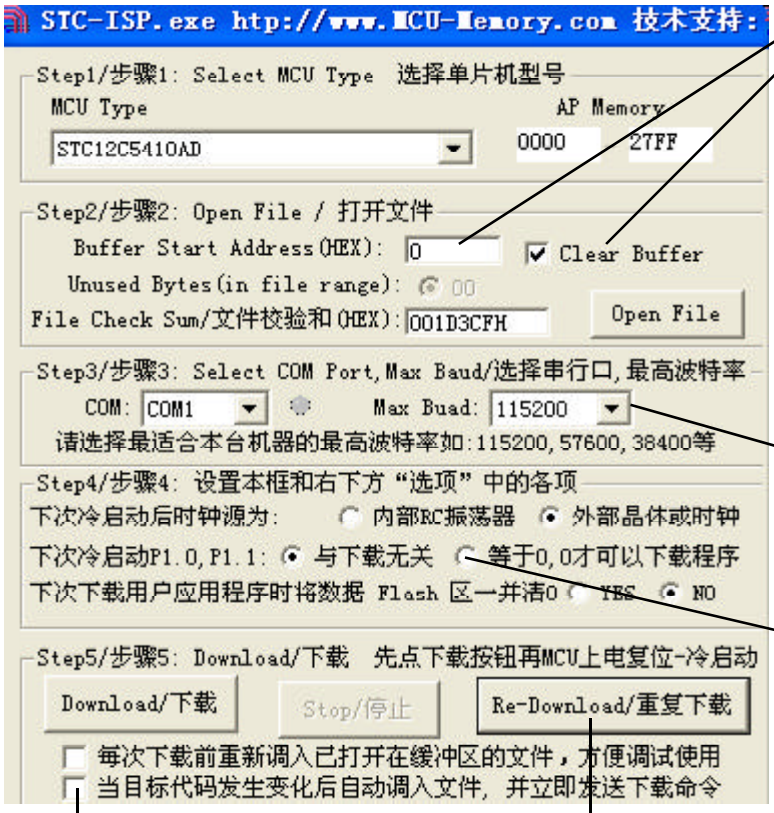
支持 *.bin,*.hex(Intel 16 进制格式)文件，少数 *.hex 文件不支持的话，请转换成 *.bin 文件

请随时注意升级 PC（电脑）端的 STC-ISP.EXE 程序。

(3).STC12系列单片机出厂时就已完全加密。需要单片机内部的电放光后上电复位(冷起动)才运行系统ISP程序，如从 P3.0/RxD 检测到合法的下载命令流就下载用户程序，如检测不到就复位到用户程序区，运行用户程序。

(4).如果用户板上 P3.0/RxD，P3.1/Txd 接了 RS-485 等电路，下载时需要将其断开。用户系统接了 RS-485 等通信电路，推荐在选项中选择“下次冷启动时需 P1.0/P1.1=0/0 才可以下载程序”

11.1.3 电脑端的 ISP 控制软件界面使用说明



第一次调文件进缓冲区，要清缓冲区。
要调几个文件进缓冲区，如EEPROM里的数据文件需要和应用程序文件一次同时ISP下载编程进单片机：
除每次均要指定缓冲区起始地址外，第二次及以后不能清缓冲区
如将要写入EEPROM区的数据文件调入从缓冲区2800H或1000H开始的地方,并不清缓冲区,然后和应用程序一起写入

用户根据实际使用效果选择限制最高通信波特率，如57600，38400，19200

如P30/P31外接RS-485/RS-232等通信电路，建议选择P10/P11等于0/0才可以下载程序，如不同时为0/0，则跨过系统ISP引导程序，直接运行用户程序。

新的设置冷启动后（彻底停电后再上电），才生效

开发调试时，可考虑选择此项

大批量生产时使用

Step1/ 步骤1：选择你所使用的单片机型号，如STC12C5410,STC12C5410AD 等
Step2/ 步骤2：打开文件，要烧录用户程序，必须调入用户的程序代码（*.bin, *.hex）
Step3/ 步骤3：选择串行口，你所使用的电脑串口，如串行口1--COM1, 串行口2--COM2,...
有些新式笔记本电脑没有RS-232串行口,可买一条USB-RS232转接器，人民币50元左右。
有些USB-RS232转接器，不能兼容，可让宏晶帮你购买经过测试的转换器。
Step4/ 步骤4：选择下次冷启动后，时钟源为"内部R/C振荡器"还是"外部晶体或时钟"。
Step5/ 步骤5：选择"Download/下载"按钮下载用户的程序进单片机内部，可重复执行
Step5/ 步骤5， 也可选择"Re-Download/重复下载"按钮
下载时注意看提示，主要看是否要给单片机上电或复位，下载速度比一般通用编程器快。
一定要先选择"Download/下载"按钮，然后再给单片机上电复位(先彻底断电)，而不要
先上电，先上电，检测不到合法的下载命令流，单片机就直接跑用户程序了。
关于硬件连接：

(1). MCU/单片机 RXD(P3.0) --- RS-232转换器 --- PC/电脑 TXD(COM Port Pin3)
(2). MCU/单片机 TXD(P3.1) --- RS-232转换器 --- PC/电脑 RXD(COM Port Pin2)
(3). MCU/单片机 GND ---------- PC/电脑 GND(COM Port Pin5)
(4). 如果您的系统P3.0/P3.1连接到 RS-485 电路，推荐
在选项里选择"下次冷启动需要P1.0/P1.1 = 0,0才可以下载用户程序"
这样冷启动后如 P1.0, P1.1 不同时为0,单片机直接运行用户程序，免得由于RS-485总线上的乱码造成单片机反复判断乱码是否为合法，浪费几百mS的时间，其实如果你的系统本身P3.0,P3.1就是做串口使用，也建议选择P1.0/P1.1 = 0/0才可下载用户程序，以便下次冷启动直接运行用户程序。
(5). RS-232转换器可选用STC232/MAX232/SP232(4.5-5.5V),STC3232/MAX3232/SP3232(3V-5.5V).
STC232/MAX232/SP232尽量选用SOP封装(窄体)，STC3232尽量选用SOP封装(窄体).

11.1.4 宏晶科技的 ISP 下载编程工具硬件使用说明

如用户系统没有 RS-232 接口，
可使用 STC-ISP Ver 3.0A.PCB 演示板作为编程工具

STC-ISP Ver 3.0APCB 板可以焊接 3 种电路，分别支持 STC12 系列 20Pin / 28Pin 及 STC89 系列 40Pin 的单片机。我们在下载板的反面贴了一张标签纸，说明它是支持 20Pin / 28Pin / 40Pin 中的哪一种，用户要特别注意。在正面焊的编程烧录用锁紧座都是 40Pin 的，锁紧座第 20-Pin 接的是地线，请将单片机的地线对着锁紧座的地线插。

在 STC-ISP Ver 3.0A PCB 板完成下载编程用户程序的工作：

关于硬件连接：

(1). 根据单片机的工作电压选择单片机电源电压

A. 5V 单片机,短接 JP1 的 MCU-VCC，+5V 电源管脚

B. 3V 单片机,短接 JP1 的 MCU-VCC，3.3V 电源管脚

(2).连接线(宏晶提供)

A. 将一端有 9 芯连接座的插头插入 PC/ 电脑 RS-232 串行接口插座用于通信

B. 将同一端的 USB 插头插入 PC/ 电脑 USB 接口用于取电

C. 将只有一个 USB 插头的一端插入宏晶的 STC-ISP Ver 3.0A PCB 板 USB1 插座用于 RS-232 通信和供电,此时 USB +5V Power 灯亮(D43,USB 接口有电)

(3).其他插座不需连接

(4).SW1 开关处于非按下状态，此时 MCU-VCC Power 灯不亮(D41)，没有给单片机通电

(5).SW3 开关

处于非按下状态，P1.0，P1.1 = 1，1,不短接到地。

处于按下状态， P1.0，P1.1 = 0，0,短接到地。

如果单片机已被设成“下次冷启动 P1.0/P1.1 = 0,0 才判 P3.0/RxD 有无合法下载命令流”

就必须将 SW3 开关处于按下状态，让单片机的 P1.0/P1.1 短接到地

(6).将单片机插进 U1-Socket 锁紧座，锁紧单片机,注意单片机是 20-Pin / 28-Pin,而 U1-Socket 锁紧座是 40-Pin,我们的设计是靠下插，靠近晶体的那一端插。

(7).关于软件：选择“Download/ 下载”(必须在给单片机上电之前让 PC 先发一串合法下载命令)

(8).按下 SW1 开关，给单片机上电复位，此时 MCU-VCC Power 灯亮(D41)

此时 STC 单片机进入 ISP 模式(STC12 系列冷启动进入 ISP)

(9).下载成功后，再按 SW1 开关，此时 SW1 开关处于非按下状态，MCU-VCC Power 灯不亮(D41)，给单片机断电，取下单片机，换上新的单片机。

11.1.5 用户板没有 RS-232 转换器,如何用宏晶科技的 ISP 下载板做 RS-232 通信转换

利用 STC-ISP Ver 3.0A PCB 板进行 RS-232 转换
单片机在用户自己的板上完成下载 / 烧录：

1. U1-Socket 锁紧座不得插入单片机
2. 将用户系统上的电源(MCU-VCC,GND)及单片机的 P3.0/RXD,P3.1/TXD 接入转换板 CN2 插座
   这样用户系统上的单片机就具备了与PC/电脑进行通信的能力
3. 将用户系统的单片机的 P1.0,P1.1 接入转换板 CN2 插座(如果需要的话)
4. 如须 P1.0，P1.1 = 0，0,短接到地，可在用户系统上将其短接到地，或将 P1.0/P1.1 也从用户系统上引到 STC-ISP Ver3.0A PCB 板上，将 SW3 开关按下，则 P1.0/P1.1=0,0。
5. 关于软件：选择“Download/ 下载”
6. 给单片机系统上电复位(注意是从用户系统自供电，不要从电脑 USB 取电,电脑 USB 座不插)
7. 下载程序时，如用户板有外部看门狗电路，不得启动，单片机必须有正确的复位,但不能在 ISP 下载程序时被外部看门狗复位,如有，可将外部看门狗电路 WDI 端 / 或 WDO 端浮空
8. 如有 RS-485 晶片连到 P3.0/Rxd,P3.1/Txd,或其他线路，在下载时应将其断开。

11.2 编译器/汇编器，编程器，仿真器

STC 单片机应使用何种编译器 / 汇编器：

1.任何老的编译器 / 汇编器都可以支持，流行用 Keil C51

2.把 STC 单片机，当成 Intel 的 8052/87C52/87C54/87C58,Philips 的 P87C52/P87C54/P87C58 就可以了

3.如果要用到扩展的专用特殊功能寄存器，直接对该地址单元设置就行了，当然先声明特殊功能寄存器的地址较好

编程烧录器:

我们有：STC12C5410AD/STC12C2052AD 系列 ISP 经济型下载编程工具(人民币 50 元，可申请免费样品)

注意:有专门下载 28PIN/20PIN 的不同演示板，

28PIN 是 28PIN 的演示板,20PIN 是 20PIN 的演示板

仿真器:如您已有老的仿真器，可仿真普通 8052 的基本功能

STC12C5410AD/STC12C2052AD 系列单片机扩展功能如它仿不了

可以用 STC-ISP.EXE 直接下载用户程序看运行结果就可以了,如需观察变量,可自己写一小段测试程序通过串口输出到电脑端的 STC-ISP.EXE 的“串口调试助手”来显示,也很方便。

无须添加新的设备

无仿真器如何调试 / 开发用户程序

1.首先参照本手册当中的“用定时器 1 做波特率发生器”，调通串口程序，这样，要观察变量就可以自己写一小段测试程序将变量通过串口输出到电脑端的 STC-ISP.EXE 的“串口调试助手”来显示,也很方便 。

2.调通按键扫描程序(到处都有大量的参考程序)

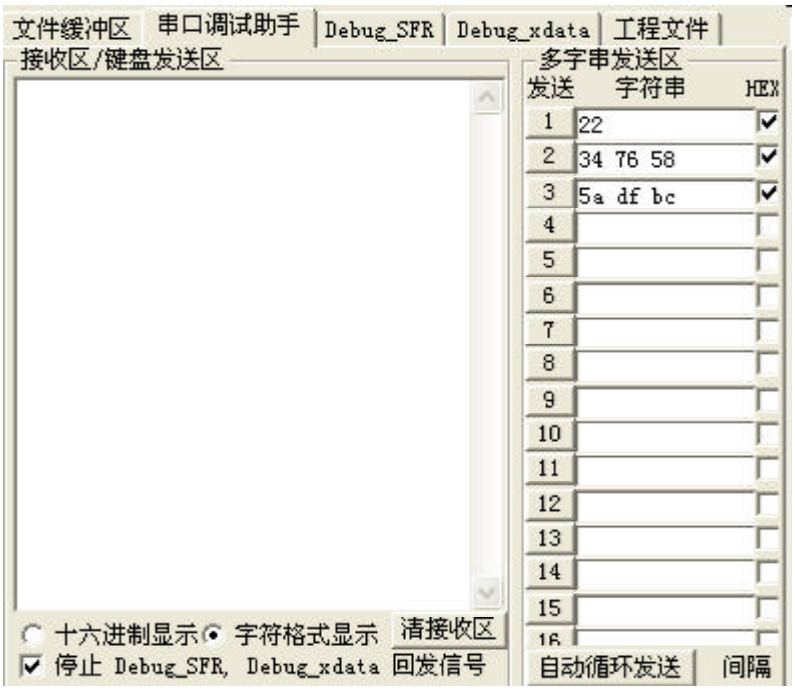
3.调通用户系统的显示电路程序，此时变量 / 寄存器也可以通过用户系统的显示电路显示了

4.调通A/D检测电路(我们用户手册里面有完整的参考程序)

5.调通PWM等电路(我们用户手册里面有完整的参考程序)

这样分步骤模块化调试用户程序，有些系统，熟练的 8051 用户，三天就可以调通了，难度不大的系统，一般一到二周就可以调通。

用户的串口输出显示程序可以在输出变量 / 寄存器的值之后，继续全速运行用户程序，也可以等待串口送来的“继续运行命令”，方可继续运行用户程序，这就相当于断点。这种断点每设置一个地方，就必须调用一次该显示寄存器 / 变量的程序，有点麻烦，但却很实用。



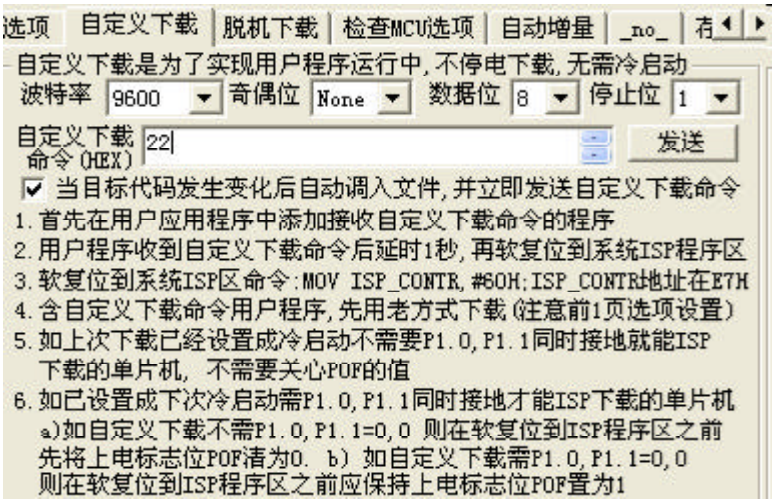
11.3 自定义下载演示程序(实现不停电下载)

```
/* ---  STC International Limited ------------------------------------------ */
/* ---  宏晶科技    姚永平    2006/7/31    V1.0 ------------------------------- */
/* ---  STC8912C5410AD  系列单片机,软件实现自定义下载程序 ------------------------ */
/* ---  Mobile: 13922805190 ------------------------------------------------- */
/* ---  Fax: 0755-82944243 -------------------------------------------------- */
/* ---  Tel: 0755-82948409 -------------------------------------------------- */
/* ---  Web: www.mcu-memory.com --------------------------------------------- */
/* ---  本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ---------------- */
/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 ----------- */
/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序----------- */
//STC12C4052, 1T 8051
#include<reg52.h>
#include<intrins.h>
sfr  ISP_CONTR =  0xE7;
sfr  CCON    = 0xD8;
sfr  CMOD    = 0xD9;
sfr  CL      = 0xE9;
sfr  CH      = 0xF9;
sfr  CCAP0L  = 0xEA;
sfr  CCAP0H  = 0xFA;
sfr  CCAPM0  = 0xDA;
sfr  CCAPM1  = 0xDB;
sbit CR      = 0xDE;
sbit MCU_Start_Led  =  P1^7;
//unsigned char self_command_array[4] =  {0x22,0x33,0x44,0x55};
#define Self_Define_ISP_Download_Command  0x22
#define RELOAD_COUNT  0xfb   //18.432MHz,12T,SMOD=0,9600bps

void serial_port_initial();
void send_UART(unsigned char);
void UART_Interrupt_Receive(void);
void soft_reset_to_ISP_Monitor(void);
void delay(void);
void display_MCU_Start_Led(void);
void send_PWM(void);

void main(void)
{
    unsigned  char i = 0;
    serial_port_initial();       // 串口初始化
    display_MCU_Start_Led();     // 点亮发光二极管表示单片机开始工作
    send_UART(0x34);             // 串口发送数据表示单片机串口正常工作
    send_UART(0xa7);             // 串口发送数据表示单片机串口正常工作
    send_PWM();                   //6kHz PWM,  50% duty
    while(1);
}
```

```
void serial_port_initial()
{
    SCON    =   0x50;   //0101,0000 8位可变波特率，无奇偶校验位
    TMOD    =   0x21;   //0011,0001 设置顶时器1为8位自动重装计数器
    TH1     =   RELOAD_COUNT;   // 设置定时器1自动重装数
    TL1     =   RELOAD_COUNT;
    TR1     =   1;     // 开定时器1
    ES      =   1;     // 允许串口中断
    EA      =   1;     // 开总中断
}

void send_UART(unsigned char i)
{
    ES     =   0;  // 关串口中断
    TI     =   0;  // 清零串口发送完成中断请求标志
    SBUF   =   i;
    while(TI ==0); // 等待发送完成
    TI     =   0;  // 清零串口发送完成中断请求标志
    ES     =   1;  // 允许串口中断
}

void UART_Interrupt_Receive(void) interrupt 4
{
    unsigned char   k   =   0;
    if(RI==1)
    {
        RI  =   0;
        k   =   SBUF;
        if(k==Self_Define_ISP_Download_Command)    // 是自定义下载命令
        {
            delay();    // 延时1秒就足够了
            delay();    // 延时1秒就足够了
            soft_reset_to_ISP_Monitor();    // 软复位到系统ISP监控区
        }
        send_UART(k);
    }
    else
    {
        TI  =  0;
    }
}

void soft_reset_to_ISP_Monitor(void)
{
    ISP_CONTR   =   0x60;   //0110,0000   软复位到系统ISP监控区
}
```

```
void delay(void)
{
    unsigned int j  =   0;
    unsigned int g  =   0;
    for(j=0;j<5;j++)
    {
        for(g=0;g<60000;g++)
        {
            _nop_();
            _nop_();
            _nop_();
            _nop_();
            _nop_();
        }
    }
}

void display_MCU_Start_Led(void)
{
    unsigned  char i = 0;
    for(i=0;i<3;i++)
    {
        MCU_Start_Led   =   0;  //顶亮MCU开始工作指示灯
        delay();
        MCU_Start_Led   =   1;  //熄灭MCU开始工作指示灯
        delay();
        MCU_Start_Led   =   0;  //顶亮MCU开始工作指示灯
    }
}

void send_PWM(void)
{
    CMOD    =   0x00;   // CIDL - - - - CPS1 CPS0 ECF  Setup PCA Timer
                        // CPS1 CPS0 = 00, Fosc/12 is PCA/PWM clock
                        // 18432000/12/256 = 6000
    CL      =   0x00;
    CH      =   0x00;
    CCAP0L  =   0x80;   //Set the initial value same as CCAP0H
    CCAP0H  =   0x80;   //50% Duty Cycle
    CCAPM0  =   0x42;   //0100,0010 Setup PCA module 0 in 8BIT PWM, P3.7
    CR      =   1;      //启动 PCA/PWM 定时器
}
```

自定义下载在 STC 的电脑端 ISP 软件 STC-ISP.EXE 中，还应做相应设置，具体参考设置见下图：



详细的帮助上图也有具体的说明

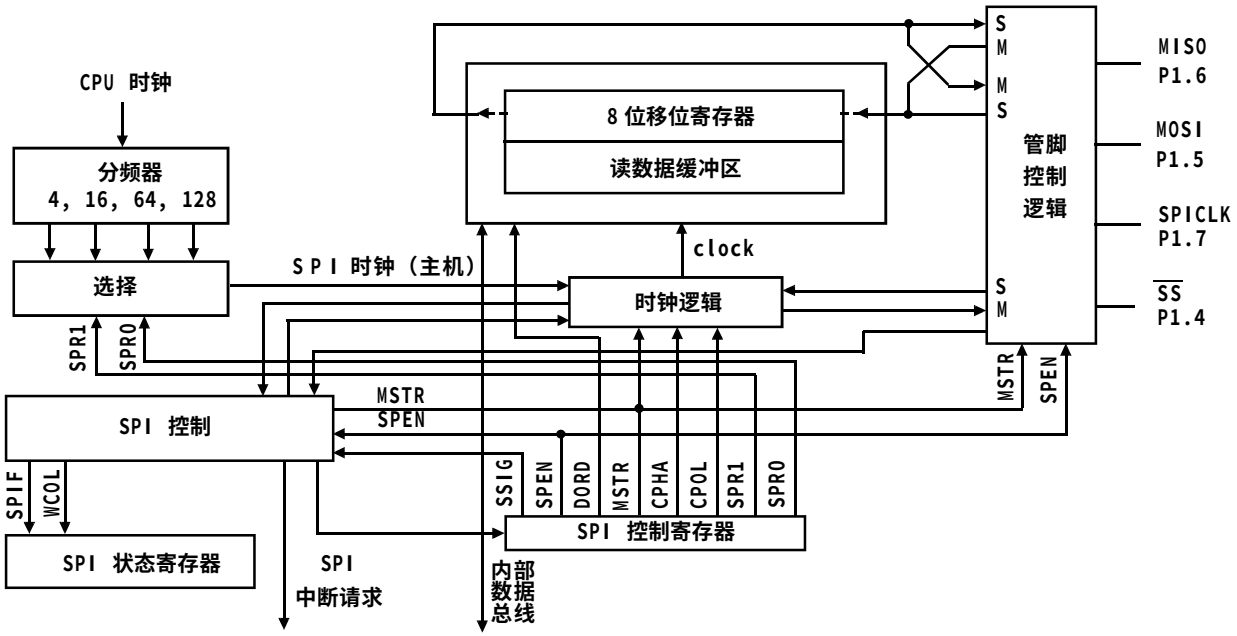
第 12 章 同步串行外围接口（SPI）及测试程序

12.1 SPI 功能模块特殊功能寄存器介绍

STC12C5410AD 及 STC12C2052AD 系列单片机还提供另一种高速串行通信接口——SPI 接口。SPI 是一种全双工、高速、同步的通信总线，有两种操作模式：主模式和从模式。在主模式中支持高达 3Mbit/s 的速率(工作频率为 12MHz 时,如果 CPU 主频采用 20MHz 到 36MHz,则可更高，从模式时速度无法太快，Fosc/8 以内较好),还具有传输完成标志和写冲突标志保护。

STC12 系列 1T 8051 单片机 SPI 功能模块特殊功能寄存器 SPI Management SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
SPCTL	85h	SPI Control Register	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	0000,0000
SPSTAT	84h	SPI Status Register	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
SPDAT	86h	SPI Data Register									0000,0000



SPI 方框图

SPI 接口有 4 个管脚：SPICLK/P1.7，MOSI/P1.5，MISO/P1.6 和 $\overline{SS}$/P1.4。

● SPICLK，MOSI 和 MISO 通常和两个或更多 SPI 器件连接在一起。数据通过 MOSI 由主机传送到从机，通过 MISO

由从机传送到主机。SPICLK 信号在主模式时为输出，在从模式时为输入。如果 SPI 系统被禁止，即 SPEN (SPCTL.6)=0(复位值)，这些管脚都可作为 I/O 口使用。

● /SS 为从机选择管脚。在典型的配置中，SPI 主机使用 I/O 口选择一个 SPI 器件作为当前的从机。SPI 从器件通过其 /SS 脚确定是否被选择。如果满足下面的条件之一，/SS 就被忽略：

－如果 SPI 系统被禁止，即 SPEN(SPCTL.6)=0(复位值)

－如果 SPI 配置为主机，即 MSTR(SPCTL.4)=1，并且 P1.4 配置为输出（通过 P1M0.4 和 P1M1.4）

－如果 /SS 脚被忽略，即 SSIG(SPCTL.7)位＝1，该脚配置用于 I/O 口功能。

注：即使 SPI 被配置为主机（MSTR ＝ 1），它仍然可以通过拉低 /SS 脚配置为从机（如果 P1.4 配置为输入且 SSIG=0)。要使能该特性，应当置位 SPIF(SPSTAT.7)。

典型连接如 SPI 图 1～3 所示。

SPI 控制寄存器的位分配（SPCTL- 地址：85h）

位	7	6	5	4	3	2	1	0
符 号	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
复 位	0	0	0	0	0	1	0	0

SPI 控制寄存器的位描述（SPCTL – 地址：85h）

位	符号	描 述
0	SPR0	SPR0/SPR1是SPI 时钟速率选择控制位。
1	SPR1	SPR1, SPR0： 0 0 －CPU_CLK/4 0 1 －CPU_CLK/16 1 0 －CPU_CLK/64 1 1 －CPU_CLK/128
2	CPHA	SPI 时钟相位选择（见SPI图4～图7）： 1：数据在SPICLK 的前时钟沿驱动，并在后时钟沿采样。 0：数据在/SS 为低（SSIG＝00）时被驱动，在SPICLK 的后时钟沿被改变，并在前时钟沿被采样。 （注：SSIG=1 时的操作未定义）
3	CPOL	SPI 时钟极性（见SPI图4～图7）： 1：SPICLK 空闲时为高电平。SPICLK 的前时钟沿为下降沿而后沿为上升沿。 0：SPICLK 空闲时为低电平。SPICLK 的前时钟沿为上升沿而后沿为下降沿。
4	MSTR	主/从模式选择（见SPI 主从选择表）。
5	DORD	SPI 数据顺序： 1：数据字的LSB(最低位) 最先发送；0：数据字的MSB(最高位) 最先发送。
6	SPEN	SPI 使能。 1：SPI 使能。 0：SPI 被禁止，所有SPI 管脚都作为I/O 口使用。
7	SSIG	/SS 忽略。 1：MSTR（位4）确定器件为主机还是从机。 0：/SS 脚用于确定器件为主机还是从机。/SS 脚可作为I/O 口使用（见SPI 主从选择表）。

SPI 状态寄存器的位分配（SPSTAT – 地址：84h）

位	7	6	5	4	3	2	1	0
符号	SPIF	WCOL	-	-	-	-	-	-
复位	0	0	X	X	X	X	X	X

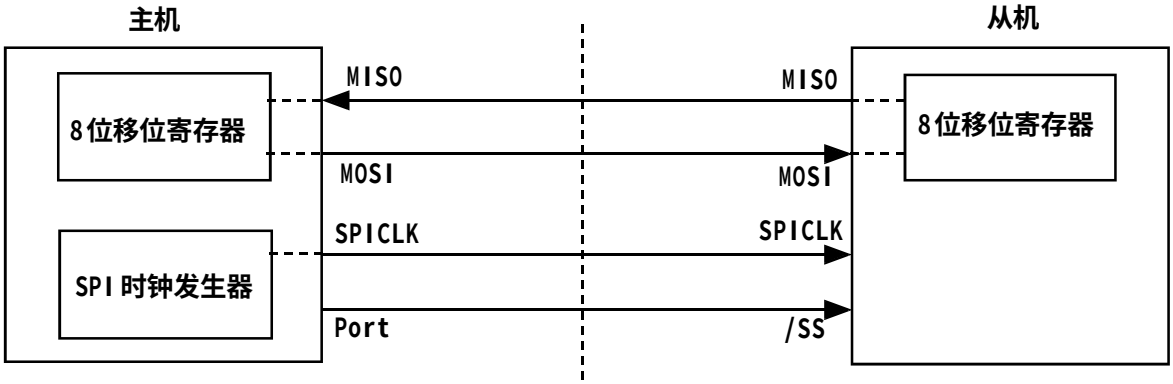
SPI 状态寄存器的位描述（SPSTAT – 地址：84h）

位	符号	符号
7	SPIF	SPI 传输完成标志。当一次串行传输完成时，SPIF 置位，并当ESPI 和EA 都置位时产生中断。当SPI 处于主模式且SSIG=0 时，如果/SS 为输入并被驱动为低电平，SPIF 也将置位。SPIF标志通过软件向其写入“1”清零。
6	WCOL	SPI 写冲突标志。在数据传输的过程中如果对SPI 数据寄存器SPDAT 执行写操作，WCOL 将置位。WCOL 标志通过软件向其写入“1”清零。
5 - 0	-	保留

SPI 数据寄存器的位分配（SPDAT – 地址：86h）

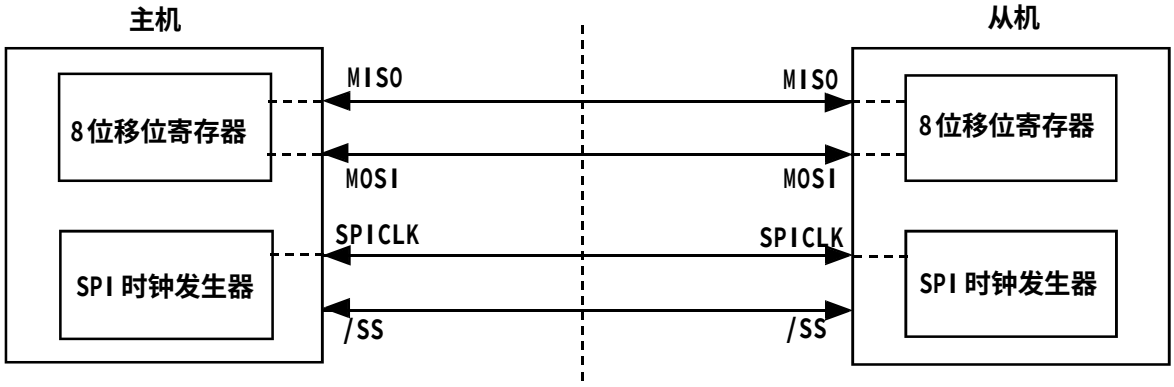
位	7	6	5	4	3	2	1	0
符 号	MSB							LSB
复 位	0	0	0	0	0	0	0	0

SPDAT.7 - SPDAT.0: 传输的数据位 Bit7～Bit0



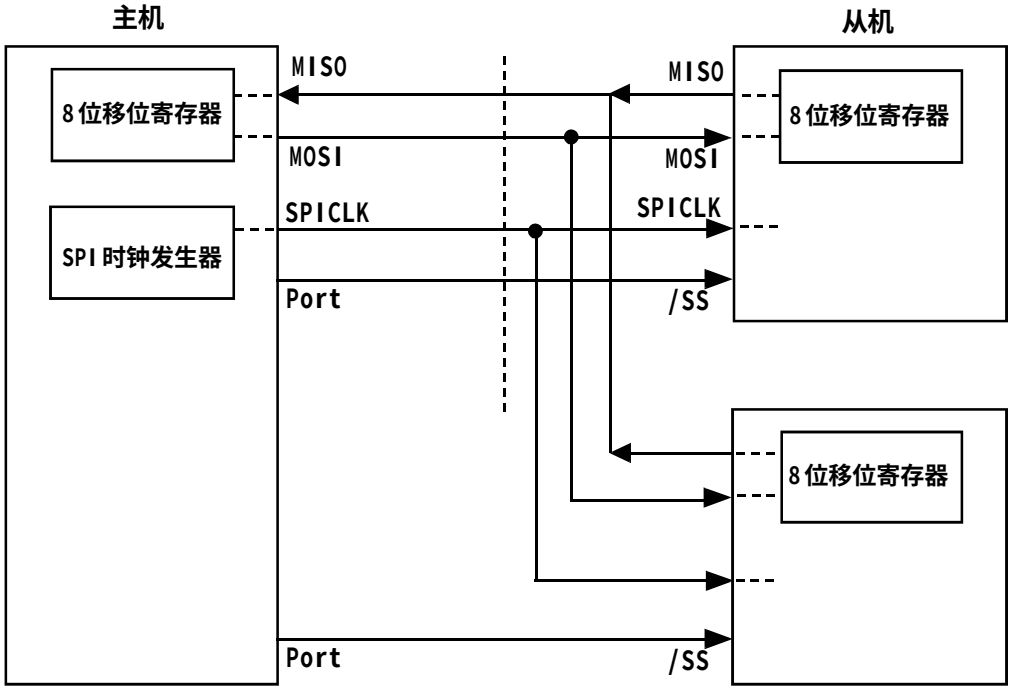
SPI 图1 SPI 单主机-单从机 配置

在上图 SPI 图 1 中，从机的 SSIG(SPCTL.7)为 0，/SS 用于选择从机。SPI 主机可使用任何端口（包括 P1.4/$\overline{SS}$）来驱动 /SS 脚。



SPI 图2 SPI 双器件配置（器件可互为主从）

上图 SPI 图 2 所示为两个器件互为主从的情况。当没有发生 SPI 操作时，两个器件都可配置为主机(MSTR=1)，将 SSIG 清零并将 P1.4(/SS)配置为准双向模式。当其中一个器件启动传输时，它可将 P1.4 配置为输出并驱动为低电平，这样就强制另一个器件变为从机。



SPI 图3 SPI 单主机-多从机 配置

在上图 SPI 图 3 中，从机的 SSIG(SPCTL.7)为 0，从机通过对应的 /SS 信号被选中。SPI 主机可使用任何端口（包括 P1.4/$\overline{SS}$）来驱动 /SS 脚。

对 SPI 进行配置

下表 所示为主 / 从模式的配置以及模式的使用和传输方向。

SPI 主从模式选择

SPEN	SSIG	/SS 脚 P1.4	MSTR	主或从模式	MISO P1.6	MOSI P1.5	SPICLK P1.7	备注
0	X	P1.4	X	SPI 功能禁止	P1.6	P1.5	P1.7	SPI 禁止。P1.4/P1.5/P1.6/P1.7作为普通I/O口使用
1	0	0	0	从机模式	输出	输入	输入	选择作为从机
1	0	1	0	从机模式未被选中	高阻	输入	输入	未被选中。MISO 为高阻状态，以避免总线冲突
1	0	0	1—>0	从机模式	输出	输入	输入	P1.4/ SS 配置为输入或准双向口。SSIG 为0。如果择/SS 被驱动为低电平，则被选择作为从机。当SS 变为低电平时，MSTR将清零。 注：当/SS处于输入模式时，如被驱动为低电平且SSIG=0 时，MSTR 位自动清零。
1	0	1	1	主(空闲)	输入	高阻	高阻	当主机空闲时MOSI 和SPICLK 为高阻态以避免总线冲突。用户必须将SPICLK 上拉或下拉（根据CPOL-SPCTL.3 的取值）以避免SPICLK出现悬浮状态。
				主(激活)		输出	输出	作为主机激活时，MOSI 和SPICLK 为推挽输出
1	1	P1.4	0	从	输出	输入	输入	
1	1	P1.4	1	主	输入	输出	输出	

作为从机时的额外注意事项

当 CPHA = 0 时，SSIG 必须为 0，/SS 脚必须取反并且在每个连续的串行字节之间重新设置为高电平。如果 SPDAT 寄存器在 /SS 有效（低电平）时执行写操作，那么将导致一个写冲突错误。CPHA=0 且 SSIG=0 时的操作未定义。

当 CPHA = 1 时，SSIG 可以置位。如果 SSIG = 0，/SS 脚可在连续传输之间保持低有效（即一直固定为低电平）。这种方式有时适用于具有单固定主机和单从机驱动 MISO 数据线的系统。

作为主机时的额外注意事项

在 SPI 中，传输总是由主机启动的。如果 SPI 使能（SPEN=1）并选择作为主机，主机对 SPI 数据寄存器的写操作将启动 SPI 时钟发生器和数据的传输。在数据写入 SPDAT 之后的半个到一个 SPI 位时间后，数据将出现在 MOSI 脚。

需要注意的是，主机可以通过将对应器件的 /SS 脚驱动为低电平实现与之通信。写入主机 SPDAT 寄存器的数据从 MOSI 脚移出发送到从机的 MOSI 脚。同时从机 SPDAT 寄存器的数据从 MISO 脚移出发送到主机的 MISO 脚。

传输完一个字节后，SPI 时钟发生器停止，传输完成标志（SPIF）置位并产生一个中断（如果 SPI 中断使能）。主机和从机 CPU 的两个移位寄存器可以看作是一个 16 循环移位寄存器。当数据从主机移位传送到从机的同时，数据也以相反的方向移入。这意味着在一个移位周期中，主机和从机的数据相互交换。

通过 /SS 改变模式

如果 SPEN=1，SSIG=0 且 MSTR=1，SPI 使能为主机模式。/SS 脚可配置为输入或准双向模式。这种情况下，另外一个主机可将该脚驱动为低电平，从而将该器件选择为 SPI 从机并向其发送数据。

为了避免争夺总线，SPI 系统执行以下动作:

1)MSTR 清零并且 CPU 变成从机。这样 SPI 就变成从机。MOSI 和 SPICLK 强制变为输入模式，而 MISO 则变为输出模式。

2）SPSTAT 的 SPIF 标志位置位。如果 SPI 中断已被使能，则产生 SPI 中断。

用户软件必须一直对 MSTR 位进行检测，如果该位被一个从机选择所清零而用户想继续将 SPI 作为主机，这时就必须重新置位 MSTR，否则就进入从机模式。

写冲突

SPI 在发送时为单缓冲，在接收时为双缓冲。这样在前一次发送尚未完成之前，不能将新的数据写入移位寄存器。当发送过程中对数据寄存器进行写操作时，WCOL 位（SPSTAT.6）将置位以指示数据冲突。在这种情况下，当前发送的数据继续发送，而新写入的数据将丢失。

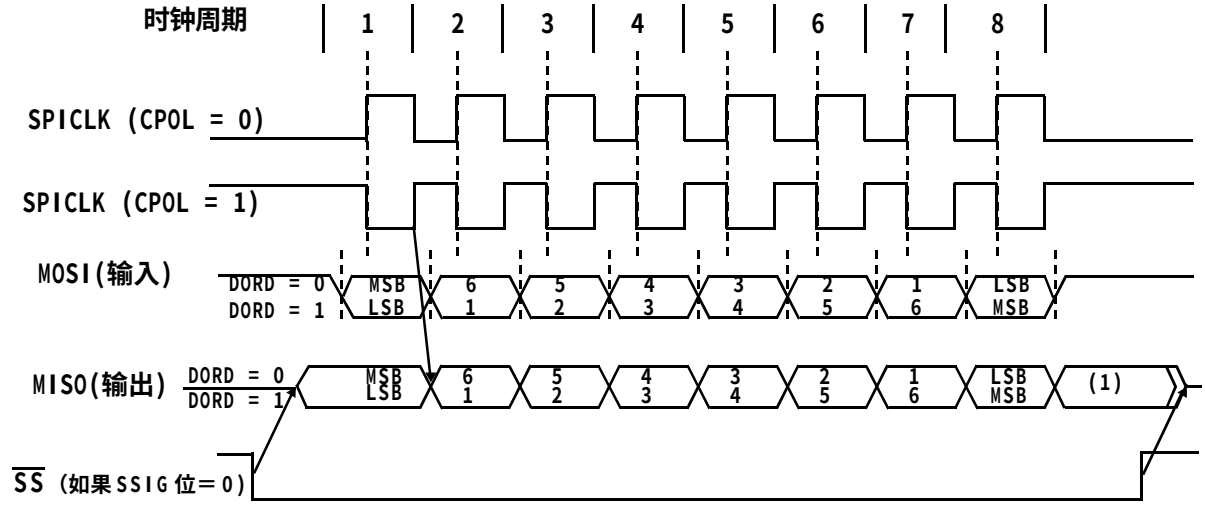
当对主机或从机进行写冲突检测时，主机发生写冲突的情况是很罕见的，因为主机拥有数据传输的完全控制权。但从机有可能发生写冲突，因为当主机启动传输时，从机无法进行控制。

接收数据时，接收到的数据传送到一个并行读数据缓冲区，这样将释放移位寄存器以进行下一个数据的接收。但必须在下个字符完全移入之前从数据寄存器中读出接收到的数据，否则，前一个接收数据将丢失 。

WCOL 可通过软件向其写入“1”清零。

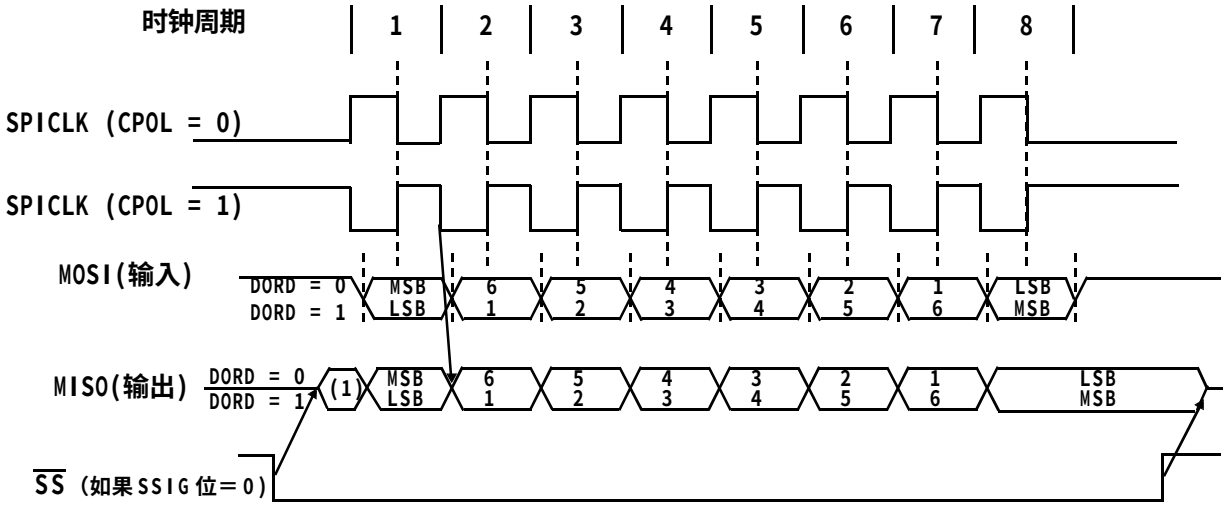
数据模式

时钟相位位(CPHA)允许用户设置采样和改变数据的时钟边沿。时钟极性位 CPOL 允许用户设置时钟极性。 SPI 图 4～图 7 所示为时钟相位位 CPHA 的不同设定。

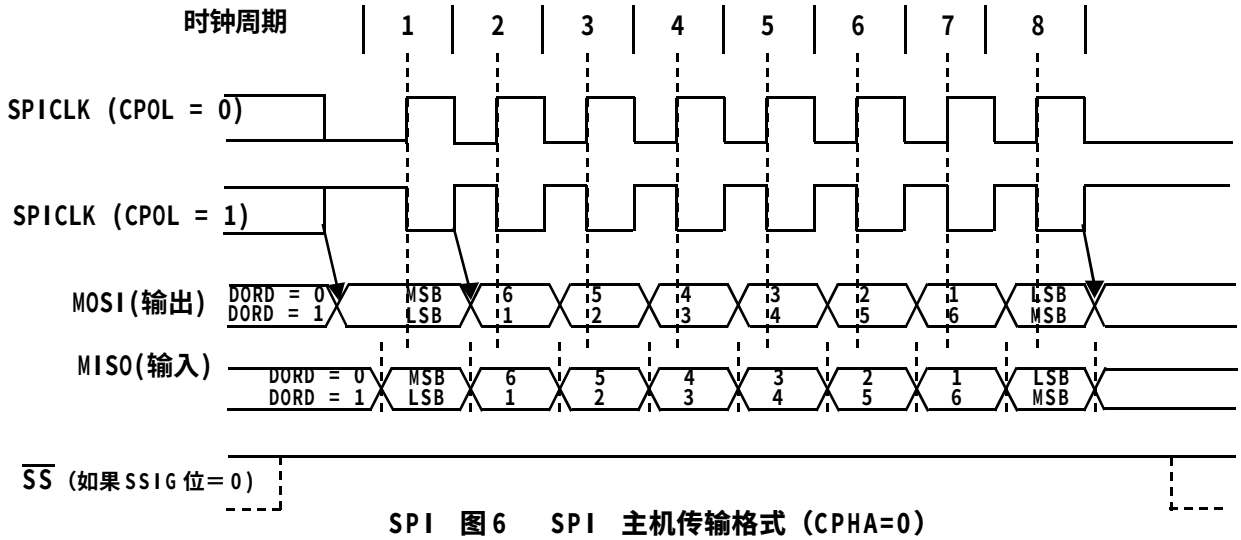


(1) ___ 未定义

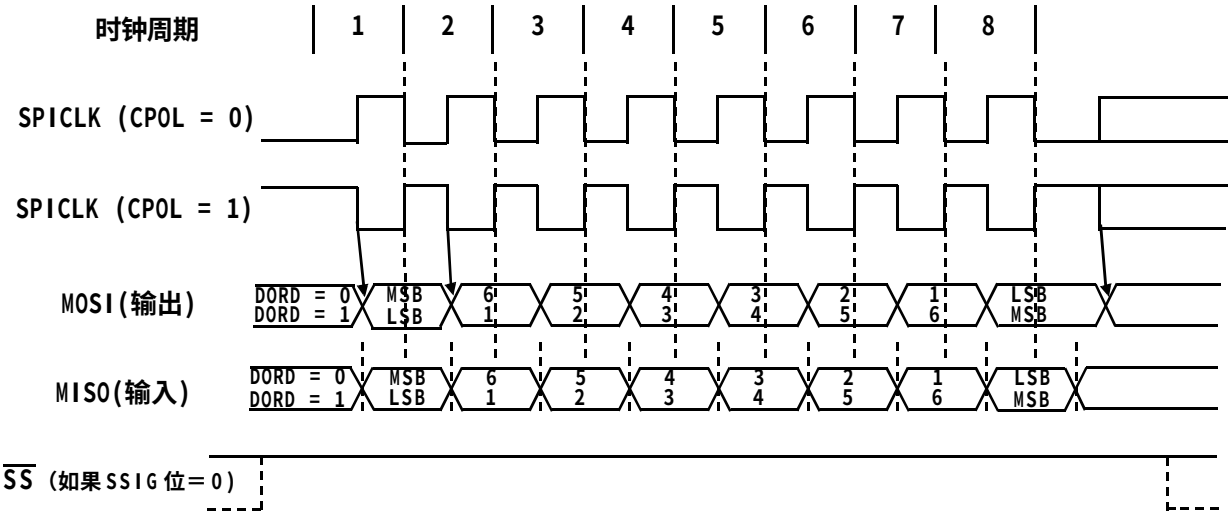
SPI 图 4 SPI 从机传输格式（CPHA=0）



SPI 图5 SPI 从机传输格式（CPHA=1）



SPI 图6 SPI 主机传输格式（CPHA=0）



SPI 图7 SPI 主机传输格式（CPHA=1）

SPI 时钟预分频器选择

SPI 时钟预分频器选择是通过 SPCTL 寄存器中的 SPR1-SPR0 位实现的

12.2 SPI 功能测试程序 1(适用于单主单从系统)

```
;/* ---  STC International Limited ------------------------------------- */
;/* ---  宏晶科技     姚永平     2006/1/6     V1.0 ---------------------------- */
;/* ---  one_master_one_slave ------------------------------------------- */
;/* ---  STC12C5412AD, STC12C5410AD, STC12C5408AD ----------------------- */
;/* ---  STC12C5406AD, STC12C5404AD, STC12C5402AD ----------------------- */
;/* ---  STC12C5052AD, STC12C4052AD, STC12C3052AD ----------------------- */
;/* ---  STC12C2052AD, STC12C1052AD, STC12C0552AD ----------------------- */
;/* ---  Mobile: 13922805190 -------------------------------------------- */
;/* ---  Fax: 0755-82944243 --------------------------------------------- */
;/* ---  Tel: 0755-82948409 --------------------------------------------- */
;/* ---  Web: www.mcu-memory.com ---------------------------------------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 ------ */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序------ */
;-------------------------------------------------------------------------
;1. 本示例程序演示 STC12C2052AD 和 STC12C5410AD 系列 MCU 的 SPI 功能，适用于
;   单主单从系统
;-------------------------------------------------------------------------
;2. 硬件连接:三线连接
;
;          主单片机       I/O口                    I/O口     从单片机
;
;      +--------------+  MISO    <-- 位流方向     MISO +--------------+
;      |      SPI     |<<----------------------------|      SPI     |
;      |8位移位寄存器 |                              |8位移位寄存器 |
;      |              |------------------------------>>|              |
;      +-------+------+  MOSI    位流方向 -->    MOSI +-------^------+
;              |                                             |
;              |         SCLK                     SCLK       |
;              +--------------------->>--------------------->+
;
;    除此之外，主单片机的 RS-232 串行口通过 RS-232 转换器与 PC 机的 RS-232
;串行口相连接。
;
;3. SPI 通讯过程：
;    主单片机与从单片机的 SPI 8位移位寄存器连接成一个循环的 16 位移位寄存器。
;当主单片机程序向 SPDAT 写入一个字节时，立即启动一个连续的 8 位移位通讯过程:
;主单片机的 SCLK 脚向从单片机的 SCLK 脚发出一串脉冲，在这串脉冲的驱动下，主
;单片机 SPI 8位移位寄存器中的数据移到了从单片机的 SPI 8位移位寄存器中；与此
;同时，从单片机 SPI 8位移位寄存器中的数据移到了主单片机的 SPI 8位移位寄存器
;中。利用这样的数据交换机制，主单片机既可向从单片机发送数据，又可读从单片机
;中的数据。
;
;4. 使用方法
;  a) 修改程序，使 MASTER EQU 1 的那行有效。汇编后的程序代码下载到主单片机中。
;  b) 修改程序，使 MASTER EQU 0 的那行有效。汇编后的程序代码下载到从单片机中。
```

```
;  c) 给主、从单片机上电。
;  d) 用串口调试助手(STC 的 ISP 下载程序 STC-ISP.exe 3.2 以上版本提供了该功能)
;       向主单片机发送一串数据。
;           主单片机的 RS-232 串口每收到一个字节就立刻将收到的字节通过 SPI 口
;       发送到从单片机中，与此同时主单片机会收到从单片机发回的一个字节(见 3. SPI
;       通讯过程)，主单片机又立刻把这个字节通过 RS-232 口发送到 PC 机。
;           从单片机的 SPI 口收到的数据后，把收到的数据放到自己的 SPDAT 寄存器
;       中，当下一次主单片机发送一个字节时把数据发回到主单片机。
;  e) 在串口调试助手接收区观察接收的数据。
;
;5. 怎样用巡测方式接收 SPI 数据
;         本示例为中断方式接收 SPI 口数据，若想用巡测方式接收 SPI 数据可以用以下
;     几行指令实现：

;    Wait_SPI_Receive_Byte:
;         MOV A, SPSTAT                    ;判收到从 SPI 发回的数据否
;         ANL A, #80H
;         JZ  Wait_SPI_Receive_Byte        ;SPI 未收到数据，继续等待
;         MOV   A, SPDAT                   ;SPI 已收到数据，将收到的数据送累加器 A
;           ...
;
;6. 实验条件: MCU 晶振频率 Fosc = 18.432MHz, PC 机 RS232 串口波特率等于 57600
;    实验结果: SPI 口传输数据无误。
;         由于本程序的 RS232 接收，SPI 端口的接收都没有使用接收缓冲区，所以 RS232
;    串口波特率不要高于 57600，若使用接收缓冲区，波特率可以到 115200 以上。
;-------------------------------------------------------------------------------
;定义常量
;-----------------
;定义功能常量，以下两行注释其中一行，取消另一行注释使之有效
;MASTER    EQU 1             ;汇编后的程序代码下载到主单片机中
MASTER    EQU 0             ;汇编后的程序代码下载到从单片机中
;-----------------
;定义波特率自动重装数常量
;以下波特率是 PCON.7 = 0 时的数值，若使 PCON.7 = 1 可将波特率加倍
;RELOAD_8BIT_DATA   EQU 0FFH    ;Fosc=22.1184MHz, Baud = 57600
;RELOAD_8BIT_DATA   EQU 0FBH    ;Fosc=18.432MHz, Baud=9600, 1T 运行时 Baud=115200
RELOAD_8BIT_DATA    EQU 0F6H    ;Fosc=18.432MHz, Baud=4800, 1T 运行时 Baud=57600
;RELOAD_8BIT_DATA   EQU 0FFH    ;Fosc=11.059MHz, Baud = 28800、
;-------------------------------------------------------------------------------
;定义特殊功能寄存器
AUXR     EQU 8EH
;AUXR 特殊功能寄存器的 bit3 是 SPI 中断允许控制位 ESPI
;IE 特殊功能寄存器的 bit5 是 ADC 和 SPI 两个中断共享的总中断允许控制位 EADC_SPI
;要产生 SPI 中断，需要 ESPI/EADC_SPI/EA 都为 1
;-------------------------------------------------------------------------------
```

```
;定义 SPI 特殊功能寄存器，详细说明见本程序的后部或 STC 12C5410AD 中文指南
SPCTL   EQU 85H
SPSTAT  EQU 84H
SPDAT   EQU 86H

EADC_SPI EQU IE.5
;-------------------------------------------------------------------------------
;定义 SPI 脚
SCLK    EQU P1.7
MISO    EQU P1.6
MOSI    EQU P1.5
SS      EQU P1.4
;-------------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START       EQU P3.4
;-------------------------------------------------------------------------------
;定义变量
Flags       EQU 20H
SPI_Receive EQU Flags.0             ;SPI 端口收到数据标志位
SPI_buffer  EQU 30H                 ;该变量用于保存 SPI 端口收到的数据
;-------------------------------------------------------------------------------
    ORG   0000H
    LJMP  MAIN
;-------------------------------------------------------------------------------
    ORG   002BH                     ;ADC_SPI 中断服务程序入口
    LJMP  ADC_SPI_Interrupt_Routine
;-------------------------------------------------------------------------------
    ORG   0080H
MAIN:
    CLR   LED_MCU_START             ;点亮 MCU 开始工作指示灯
    MOV   SP, #7FH
    ACALL Init_System               ;系统初始化
if MASTER
Check_RS232:
        JNB   RI, Master_Check_SPI  ;判 RS-232 串口中收到数据否
        ;主单片机 RS-232 串口已收到新的数据
        ACALL Get_Byte_From_RS232   ;主单片机将 RS-232 串口中收到的数据送到累加器 A
        ACALL SPI_Send_Byte         ;主单片机将累加器 A 中的数据发送到从机 SPI
        SJMP  Check_RS232
Master_Check_SPI:
        JNB   SPI_Receive, Check_RS232 ;判收到从 SPI 发回的数据否
        ;主单片机 SPI 端口已收到新的数据
        MOV   A, SPI_buffer         ;将 "从 SPI 发回的数据"送到累加器 A
        CLR   SPI_Receive           ;清 0 主单片机 SPI 端口收到数据标志位
        ACALL RS232_Send_Byte       ;将累加器 A 中的数据发送到 PC 机
        SJMP  Check_RS232
```

```
else
Slave_Check_SPI:
        JNB   SPI_Receive, Slave_Check_SPI ;判收到主 SPI 发回的数据否
        ;从单片机 SPI 端口已收到新的数据
        MOV   A, SPI_buffer          ;取 "主单片机 SPI 端口发的数据"
        CLR   SPI_Receive            ;清 0 从单片机 SPI 端口收到数据标志位
        MOV   SPDAT, A               ;将收到数据送 SPDAT，准备下一次通讯时发回
        SJMP  Slave_Check_SPI
endif
;-------------------------------------------------------------------------
ADC_SPI_Interrupt_Routine:           ;ADC_SPI 中断服务程序
    ;SPI 中断服务程序
    MOV   SPSTAT, #11000000B         ;0C0H，清 0 标志位 SPIF 和 WCOL
                  ;特别注意：是向标志位 SPIF/WCOL 写 1，将 SPIF/WCOL 清成 0
                  ;特别注意：不是向标志位 SPIF/WCOL 写 0，将 SPIF/WCOL 清成 0
    MOV   A, SPDAT                   ;保存收到的数据
    MOV   SPI_buffer, A
    SETB  SPI_Receive                ;树立 SPI 端口收到数据标志
    RETI
;-------------------------------------------------------------------------
Init_System:
    ACALL Initial_UART               ;初始化串口
    ACALL Initial_SPI                ;初始化 SPI
    MOV   Flags, #0                  ;清标志字
    SETB  EA                         ;开总中断
    RET
;-------------------------------------------------------------------------
Initial_UART:                        ;初始化串口
; SCON  Bit:   7       6     5     4     3     2     1    0
;            SM0/FE   SM1   SM2   REN   TB8   RB8   TI   RI
    MOV   SCON, #50H                 ;0101,0000 8 位可变波特率，无奇偶校验

    MOV   TMOD, #21H                 ;T1 为自动重装模式
    MOV   TH1, #RELOAD_8BIT_DATA
    MOV   TL1, #RELOAD_8BIT_DATA

;    MOV   PCON, #80H                 ;取消本行指令注释，波特率加倍。
;使以下两行有效，波特率快 12 倍，即波特率 = 4800*12=57600
    MOV   A, #01000000B              ;T1 以 1T 的速度计数，是普通 8051 的 12 倍
    ORL   AUXR, A

    SETB  TR1                        ;启动定时器 1 开始计数
    RET
;-------------------------------------------------------------------------
Initial_SPI:                         ;初始化 SPI
;SPI 控制寄存器
```

```
;          7      6      5      4      3      2      1      0
;SPCTL   SSIG   SPEN   DORD   MSTR   CPOL   CPHA   SPR1   SPR0

if MASTER
    MOV   SPCTL,#11111100B          ;0FCH，忽略 SS 脚，设为主机
    ;SSIG=1: 忽略SS脚
    ;SPEN=1：允许 SPI 工作
    ;DORD=1：先传低位 LSB
    ;MSTR=1：设为主机

    ;CPOL=1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
    ;CPHA=1：数据由 SPICLK 前跳变沿驱动到 SPI 口线，SPI 模块在后跳变沿采样数据。
    ;SPR1，SPR0 = 00：主模式时 SPI 时钟源选择为 fosc/4
else
    MOV   SPCTL,#11101100B          ;0ECH，忽略 SS 脚，设为从机
    ;SSIG=1: 忽略SS脚
    ;SPEN=1：允许 SPI 工作
    ;DORD=1：先传低位 LSB
    ;MSTR=0：设为从机

    ;CPOL=1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
    ;CPHA=1：数据由 SPICLK 前跳变沿驱动到 SPI 口线，SPI 模块在后跳变沿采样数据。
    ;SPR1，SPR0 = 00：主模式时 SPI 时钟源选择为 fosc/4
endif
    MOV   SPSTAT,#11000000B         ;清0 标志位 SPIF(SPSTAT.7)，WCOL(SPSTAT.6)
                                    ;向该两个标志位写"1" 会将它们清0
    MOV   A，#00001000B
    ORL   AUXR，A                   ;令ESPI(AUXR.3)=1,允许SPIF(SPSTAT.7)产生中断
    SETB  EADC_SPI                  ;开 ADC 中断和 SPI 中断共享的总中断控制位
    RET
;-------------------------------------------------------------------------------
RS232_Send_Byte:                    ;RS232 串口发送一个字节
    CLR   TI                        ;清零串口发送中断标志
    MOV   SBUF，A
RS232_Send_Wait:
    JNB   TI，RS232_Send_Wait       ;等待发送完毕，未发送完毕跳回本行
    CLR   TI                        ;清零串口发送中断标志
    RET
;-------------------------------------------------------------------------------
;此段程序只有主 MCU 调用
SPI_Send_Byte:                      ;SPI 发送一个字节
    CLR   EADC_SPI                  ;关 ADC 中断和 SPI 中断共享的总中断控制位
    MOV   SPDAT，A                  ;SPI 发送数据
SPI_Send_Byte_Wait:
    MOV   A，SPSTAT                 ;等待 SPIF=1 即等待 SPI 发送完毕
    ANL   A，#80H
```

```
    JZ    SPI_Send_Byte_Wait
    SETB  EADC_SPI                ;开 ADC 中断和 SPI 中断共享的总中断控制位
    RET
;-------------------------------------------------------------------------------
Get_Byte_From_RS232:              ;取 RS-232 串口中收到的数据送累加器 A
    MOV   A, SBUF
    CLR   RI
    RET
;-------------------------------------------------------------------------------
    END
;-------------------------------------------------------------------------------
;更详细的资料可以参阅 STC12C5410AD.pdf (中文使用说明)。
;
;SPI 控制寄存器
;          7      6      5      4      3      2      1      0
;SPCTL   SSIG   SPEN   DORD   MSTR   CPOL   CPHA   SPR1   SPR0
;
;SSIG: 忽略 SS 脚，如果 SSIG=1，由 MSTR 位决定 SPI 主模式或从模式，
;      如果 SSIG=0，由 SS 脚决定 SPI 主模式或从模式。
;SPEN：SPI 使能位。如果 SPEN=0，SPI 功能被禁止，SPI 脚用作普通 IO 口
;DORD：SPI 数据传输顺序。
;      1：先传低位 LSB
;      0：先传高位 MSB
;MSTR：SPI 主 / 从模式选择位
;CPOL：SPI 时钟信号极性选择位
;       1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
;       0：SPI 空闲时 SPICLK = 0，前跳变沿是上升沿，后跳变沿是下降沿。
;CPHA：SPI 时钟信号相位选择位
;      1：数据由 SPICLK 前跳变沿驱动到 SPI 口线，SPI 模块在后跳变沿采样数据。
;      0：当 SS 脚为低(SSIG=0)时数据被驱动到口线，并且在 SPICLK 后跳变沿数据
;          被改变(被驱动到口线)，在 SPICLK 前跳变沿数据被采样。注意：SSIG = 1
;          时操作未定义。
;SPR1-SPR0：主模式时 SPI 时钟源选择
;      00：fosc/4
;      01：fosc/16
;      10：fosc/64
;      11：fosc/128
;
;    当 CPHA=0，SSIG 必须等于零并且在传输时 SS 脚也必须一直保持为低。当 SS 有效
;(=0)时向 SPDATA 寄存器写数据就会发生写冲突错误，WCOL 标志被置 1。
;    当 CPHA=1，SSIG 可以等于 0 或 1。如果 SSIG=0，SS 脚在连续的传输时为 0(可以
;一直保持为 0)。当系统中只有一个主和一个从 SPI 时，这是首选配置。
;-------------------
;SPI 状态寄存器
;          7      6      5      4      3      2      1      0
;SPSTAT  SPIF   WCOL     -      -      -      -      -      -
```

```
;SPIF：SPI 传输结束标志。当一次传输结束时， SPIF 被置 1，如果 SPI 中断被打开：
;       ESPI(AUXR.3)=1，EADC_SPI(IE.5)=1，EA(IE.7)=1，就引起中断。如果原来 SPI
;       由 SS 脚确定为是主模式(SSIG=0,SS=1)，当 SS 变成 0 时，SPIF 也会被置 1，
;       表示"模式改变"。向 SPIF 位写 1 将该标志清 0。
;WCOL：SPI 写冲突标志。当一个数据还在传输时，又向数据寄存器 SPDAT 写入数据，WCOL
;       就会被被置 1。向 WCOL 位写 1 将该标志清 0。
;--------------------
;SPI 主/从模式选择
;
;SPEN SSIG SS MSTR    模式    MISO   MOSI SPICLK     注释
;  0    X    X    X    禁止 SPI   输入   输入   输入     禁止 SPI 功能
;  1    0    0    0       从     输出   输入   输入     被选为从
;  1    0    1    0     未选从   输入   输入   输入     从，但没有被选中
;  1    0    0  1->0     从     输出   输入   输入     由主变为从
;  1    0    1    1       主     输入   输出   输出
;  1    1    X    0       从     输出   输入   输入     从
;  1    1    X    1       主     输入   输出   输出     主
```

12.3 SPI 功能测试程序 2(适用于单主多从系统)

```
;/* ---  STC International Limited ------------------------------------------ */
;/* ---  宏晶科技    姚永平    2006/1/6    V1.0 ------------------------------ */
;/* ---  one_master_more_slave ---------------------------------------------- */
;/* ---  STC12C5412AD, STC12C5410AD, STC12C5408AD --------------------------- */
;/* ---  STC12C5406AD, STC12C5404AD, STC12C5402AD --------------------------- */
;/* ---  STC12C5052AD, STC12C4052AD, STC12C3052AD --------------------------- */
;/* ---  STC12C2052AD, STC12C1052AD, STC12C0552AD --------------------------- */
;/* ---  Mobile: 13922805190 ------------------------------------------------ */
;/* ---  Fax: 0755-82944243 ------------------------------------------------- */
;/* ---  Tel: 0755-82948409 ------------------------------------------------- */
;/* ---  Web: www.mcu-memory.com -------------------------------------------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 ------  */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序------- */

;-------------------------------------------------------------------------------
;1. 本示例程序演示 STC12C2052AD 和 STC12C5410AD 系列 MCU 的 SPI 功能，适用于
;   单主多从系统
;-------------------------------------------------------------------------------
;2. 硬件连接:
;
;          主单片机                                    从单片机 #1
;      +--------------+    <-- 位流方向               +--------------+
;      |         MISO |<<-------------------*---------| MISO         |
;      |              |    位流方向 -->      |         |              |
;      |         MOSI |------------------*---|------->>| MOSI         |
;      |              |                  |   |         |              |
;      |         SCLK |--------------*---|---|------->>| SCLK         |
;      |              |              |   |   |         | __           |
;      |         P1.2 |--------------|---|---|------->>| SS           |
;      |              |              |   |   |         |              |
;      |              |              |   |   |         +--------------+
;      |              |              |   |   |
;      |              |              |   |   |           从单片机 #2
;      |              |              |   |   |         +--------------+
;      |              |              |   |   <<--------| MISO         |
;      |              |              |   |             |              |
;      |              |              |   +------------>>| MOSI         |
;      |              |              |                 |              |
;      |              |              +---------------->>| SCLK         |
;      |              |                                | __           |
;      |         P1.3 |------------------------------->>| SS           |
;      |              |                                |              |
;      +--------------+                                +--------------+
;
;     除此之外，主单片机的 RS-232 串行口通过 RS-232 转换器与 PC 机的 RS-232
;串行口相连接。
```

```
;
;3. SPI 通讯过程：
;     主单片机与从单片机的 SPI 8位移位寄存器连接成一个循环的 16 位移位寄存器。
;当主单片机程序向 SPDAT 写入一个字节时，立即启动一个连续的 8 位移位通讯过程:
;主单片机的 SCLK 脚向从单片机的 SCLK 脚发出一串脉冲，在这串脉冲的驱动下，主
;单片机 SPI 8位移位寄存器中的数据移到了从单片机的 SPI 8位移位寄存器中; 与此
;同时，从单片机 SPI 8位移位寄存器中的数据移到了主单片机的 SPI 8位移位寄存器
;中。利用这样的数据交换机制，主单片机既可向从单片机发送数据，又可读从单片机
;中的数据。
;
;4. 使用方法
;  a) 修改程序，使 MASTER_SLAVE EQU 0 的那行有效。汇编后的程序代码下载到
;       主单片机中。
;  b) 修改程序，使 MASTER_SLAVE EQU 1 的那行有效。汇编后的程序代码下载到
;       从单片机 #1 中。
;  c) 修改程序，使 MASTER_SLAVE EQU 2 的那行有效。汇编后的程序代码下载到
;       从单片机 #2 中。
;   d) 给主、从单片机上电。
;  e) 主单片机用 Slave1_SS 和 Slave2_SS 口线选择当前选中的从单片机，每一时刻
;       只有一个从单片机被选中。当 Slave1_SS 的 LED 灯亮时，从单片机 #1 被选中;
;       当 Slave2_SS 的 LED 灯亮时，从单片机 #2 被选中。
;           用串口调试助手(STC 的 ISP 下载程序 STC-ISP.exe 3.2 以上版本提供了
;      该功能)向主单片机发送一串数据。主单片机每收到一个字节就立刻将收到的字节
;      通过 SPI 口发送到当前选中的从单片机中。从单片机 #1 将 SPI 口收到的数据
;      再放到自己的 SPDAT 寄存器中，当下一次主单片机发送一个字节时把数据发回到
;      主单片机; 从单片机 #2 将 SPI 口收到的数据加 1 以后再放到自己的 SPDAT
;       寄存器中，当下一次主单片机发送一个字节时把数据发回到主单片机。
;  f) 在串口调试助手接收区观察接收的数据。
;
;5. 用巡测方式接收 SPI 数据
;        本示例为中断方式接收 SPI 口数据，若想用巡测方式接收 SPI 数据可以用以下
;    几行指令实现：

;   Wait_SPI_Receive_Byte:
;       MOV A, SPSTAT                   ;判收到从 SPI 发回的数据?
;       ANL A, #80H
;       JZ  Wait_SPI_Receive_Byte       ;SPI 未收到数据，继续等待
;         ...                           ;SPI 已收到数据
;         ...
;
;6. 实验条件: MCU 晶振频率 Fosc = 18.432MHz, PC 机 RS232 串口波特率等于 57600
;   实验结果: SPI 口传输数据无误。
;------------------------------------------------------------------------------
;定义常量
;-----------------
;定义功能常量，以下 3 行注释其中 2 行，使一行有效
```

```
MASTER_SLAVE  EQU 0           ;汇编后的程序代码下载到主单片机中
;MASTER_SLAVE  EQU 1           ;汇编后的程序代码下载到从单片机 #1 中
;MASTER_SLAVE  EQU 2           ;汇编后的程序代码下载到从单片机 #2 中
;----------------
;定义波特率自动重装数常量
;以下波特率是 PCON.7 = 0 时的数值，若使 PCON.7 = 1 可将波特率加倍
;RELOAD_8BIT_DATA  EQU 0FFH   ;Fosc=22.1184MHz，Baud = 57600
;RELOAD_8BIT_DATA  EQU 0FBH   ;Fosc=18.432MHz，Baud=9600，1T 运行时 Baud=115200
RELOAD_8BIT_DATA   EQU 0F6H   ;Fosc=18.432MHz，Baud=4800，1T 运行时 Baud=57600
;RELOAD_8BIT_DATA  EQU 0FFH   ;Fosc=11.059MHz，Baud = 28800、
;------------------------------------------------------------------------------
;定义特殊功能寄存器
AUXR    EQU 8EH
;AUXR 特殊功能寄存器的 bit3 是 SPI 中断允许控制位 ESPI
;IE 特殊功能寄存器的 bit5 是 ADC 和 SPI 两个中断共享的总中断允许控制位 EADC_SPI
;要产生 SPI 中断，需要 ESPI/EADC_SPI/EA 都为 1
;------------------------------------------------------------------------------
;定义 SPI 特殊功能寄存器，详细说明见本程序的后部
SPCTL           EQU 85H
SPSTAT          EQU 84H
SPDAT           EQU 86H

EADC_SPI        EQU IE.5
;------------------------------------------------------------------------------
;定义 SPI 脚
SCLK            EQU P1.7
MISO            EQU P1.6
MOSI            EQU P1.5
SS              EQU P1.4

Slave1_SS       EQU P1.2
Slave2_SS       EQU P1.3
;------------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START   EQU P3.4
;------------------------------------------------------------------------------
;定义变量
Flags           EQU 20H
SPI_Receive     EQU Flags.0         ;SPI 端口收到数据标志位

T0_10mS_count   EQU 30H             ;该变量用于保存 10 毫秒计数(T0 中断次数)
SPI_buffer      EQU 31H             ;该变量用于保存 SPI 端口收到的数据
;------------------------------------------------------------------------------
    ORG   0000H
    AJMP  MAIN
;------------------------------------------------------------------------------
```

```
    ORG   000BH                     ;定时器 0 中断服务程序入口
    AJMP  timer0_Routine
;------------------------------------------------------------------------------
    ORG   002BH                     ;ADC_SPI 中断服务程序入口
    AJMP  ADC_SPI_Interrupt_Routine
;------------------------------------------------------------------------------
    ORG   0080H
MAIN:
    CLR   LED_MCU_START             ;点亮 MCU 开始工作指示灯
    MOV   SP, #7FH
    ACALL Initial_System            ;系统初始化
if MASTER_SLAVE == 0
        CLR   Slave1_SS             ;选择从单片机 #1 为当前的从单片机
Check_RS232:
        JNB   RI, Master_Check_SPI  ;判 RS-232 串口中收到数据否
        ;主单片机 RS-232 串口已收到新的数据
        ACALL Get_Byte_From_RS232   ;主单片机将 RS-232 串口中收到的数据送到累加器 A
;        ACALL RS232_Send_Byte      ;调试用，将累加器 A 中的数据发送到 PC 机
;        SJMP  Check_RS232          ;调试用
        ACALL SPI_Send_Byte         ;主单片机将累加器 A 中的数据发送到从机 SPI
        SJMP  Check_RS232
Master_Check_SPI:
        JNB   SPI_Receive, Check_RS232 ;判收到从 SPI 发回的数据否
        ;主单片机 SPI 端口已收到新的数据
        MOV   A, SPI_buffer         ;将 "从 SPI 发回的数据"送到累加器 A
        CLR   SPI_Receive           ;清 0 主单片机 SPI 端口收到数据标志位
        ACALL RS232_Send_Byte       ;将累加器 A 中的数据发送到 PC 机
        SJMP  Check_RS232
else
Slave_Check_SPI:
        JNB   SPI_Receive, Slave_Check_SPI ;判收到主 SPI 发回的数据否
        ;从单片机 SPI 端口已收到新的数据
        MOV   A, SPI_buffer         ;取 "主单片机 SPI 端口发的数据"
        CLR   SPI_Receive           ;清 0 从单片机 SPI 端口收到数据标志位
    if MASTER_SLAVE == 2
        ADD  A, #1                  ;如果是从单片机 #2，就把收到的数据加 1
    endif
        MOV   SPDAT, A              ;将收到数据送 SPDAT，准备下一次通讯时发回
        SJMP  Slave_Check_SPI
endif
;------------------------------------------------------------------------------
if MASTER_SLAVE == 0
timer0_Routine:
        PUSH  PSW                   ;保存断点现场
        PUSH  ACC
```

```
        MOV   TH0, #0C4H            ;重装数 = 65536-15360 = 50176 = C400H
                                    ;晶振频率=18.432MHz 时， 每 10mS 中断 1 次
        INC   T0_10mS_count         ;10 毫秒计数(T0 中断次数) + 1
        MOV   A, #0C7H               ;0C8H = 199，检测是否中断了 200 次（2 秒）
        CLR   C
        SUBB  A, T0_10mS_count
        JNC   timer0_Exit
        CPL   Slave1_SS             ;改变当前选择的从单片机
        CPL   Slave2_SS
        MOV   T0_10mS_count, #0     ;清 0 10 毫秒计数(T0 中断次数)

timer0_Exit:
        POP   ACC                   ;恢复断点现场
        POP   PSW
        RETI
else
timer0_Routine:                     ;本程序中从单片机不需要使用定时器 0
        RETI
endif

;-------------------------------------------------------------------------------
ADC_SPI_Interrupt_Routine:          ;ADC_SPI 中断服务程序
    ;SPI 中断服务程序
    MOV   SPSTAT, #11000000B        ;0C0H，清 0 标志位 SPIF 和 WCOL
                ;特别注意：是向标志位 SPIF/WCOL 写 1，将 SPIF/WCOL 清成 0
                ;特别注意：不是向标志位 SPIF/WCOL 写 0，将 SPIF/WCOL 清成 0
    MOV   A, SPDAT                  ;保存收到的数据
    MOV   SPI_buffer, A
    SETB  SPI_Receive               ;树立 SPI 端口收到数据标志
    RETI
;-------------------------------------------------------------------------------
Initial_System:
    ACALL Initial_UART              ;初始化串口
    ACALL Initial_SPI               ;初始化 SPI

    SETB  TR0                       ;启动 T0
    SETB  ET0                       ;开 T0 中断

    MOV   Flags, #0                 ;清标志字
    SETB  EA                        ;开总中断
    RET
;-------------------------------------------------------------------------------
Initial_UART:                       ;初始化串口
; SCON  Bit:   7       6     5     4     3     2     1    0
;            SM0/FE   SM1   SM2   REN   TB8   RB8    Tl   Rl
    MOV   SCON, #50H                ;0101,0000 8 位可变波特率，无奇偶校验
```

```
    MOV   TMOD, #21H                ;T1 为自动重装模式
    MOV   TH1, #RELOAD_8BIT_DATA
    MOV   TL1, #RELOAD_8BIT_DATA
;   MOV   PCON, #80H                ;取消本行指令注释，波特率加倍。

    ;使以下两行有效，波特率快 12 倍，即波特率 = 4800*12=57600
    MOV   A, #01000000B             ;T1 以 1T 的速度计数，是普通 8051 的 12 倍
    ORL   AUXR, A

    SETB  TR1                       ;启动定时器 1 开始计数
    RET
;-------------------------------------------------------------------------------
Initial_SPI:                        ;初始化 SPI
if MASTER_SLAVE == 0
    MOV   SPCTL,#11111100B          ;0FCH，忽略 SS 脚，设为主机
    ;SSIG=1: 忽略 SS 脚
    ;SPEN=1：允许 SPI 工作
    ;DORD=1：先传低位 LSB
    ;MSTR=1：设为主机

    ;CPOL=1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
    ;CPHA=1：数据由 SPICLK 前跳变沿驱动到 SPI 口线，SPI 模块在后跳变沿采样数据。
    ;SPR1，SPR0 = 00：主模式时 SPI 时钟源选择为 fosc/4
else
    MOV   SPCTL,#01101100B          ;6CH，设为从机，由 SS 脚决定是否已被选中
    ;SSIG=0: 由 SS 脚决定主模式或从模式。
    ;SPEN=1：允许 SPI 工作
    ;DORD=1：先传低位 LSB
    ;MSTR=0：设为从机

    ;CPOL=1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
    ;CPHA=1：数据由 SPICLK 前跳变沿驱动到 SPI 口线，SPI 模块在后跳变沿采样数据。
    ;SPR1，SPR0 = 00：主模式时 SPI 时钟源选择为 fosc/4
endif
    MOV   SPSTAT,#11000000B         ;清 0 标志位 SPIF(SPSTAT.7)，WCOL(SPSTAT.6)
                                    ;向该两个标志位写 "1" 会将它们清 0
    MOV   A, #00001000B
    ORL   AUXR, A                   ;令 ESPI(AUXR.3)=1,允许 SPIF(SPSTAT.7)产生中断
    SETB  EADC_SPI                  ;开 ADC 中断和 SPI 中断共享的总中断控制位
    RET
;-------------------------------------------------------------------------------
RS232_Send_Byte:                    ;RS232 串口发送一个字节
    CLR   TI                        ;清零串口发送中断标志
    MOV   SBUF, A
RS232_Send_Wait:
    JNB   TI, RS232_Send_Wait       ;等待发送完毕，未发送完毕跳回本行
```

```
    CLR   TI                        ;清零串口发送中断标志
    RET
;-------------------------------------------------------------------------------
;此段程序只有主 MCU 调用
SPI_Send_Byte:                      ;SPI 发送一个字节
    CLR   EADC_SPI                  ;关 ADC 中断和 SPI 中断共享的总中断控制位
    MOV   SPDAT, A                  ;SPI 发送数据
SPI_Send_Byte_Wait:
    MOV   A, SPSTAT                 ;等待 SPIF=1 即等待 SPI 发送完毕
    ANL   A, #80H
    JZ    SPI_Send_Byte_Wait
    SETB  EADC_SPI                  ;开 ADC 中断和 SPI 中断共享的总中断控制位
    RET
;-------------------------------------------------------------------------------
Get_Byte_From_RS232:                ;取 RS-232 串口中收到的数据累加器 A
    MOV   A, SBUF
    CLR   RI
    RET
;-------------------------------------------------------------------------------
    END
;-------------------------------------------------------------------------------
;更详细的资料可以参阅 STC12C5410AD.pdf (中文使用说明)。
;
;SPI 控制寄存器
;          7      6      5      4      3      2      1      0
;SPCTL   SSIG   SPEN   DORD   MSTR   CPOL   CPHA   SPR1   SPR0
;
;SSIG: 忽略 SS 脚，如果 SSIG=1，由 MSTR 位决定主模式或从模式，
;      如果 SSIG=0，由 SS 脚决定主模式或从模式。
 ;SPEN：SPI 使能位。如果 SPEN=0，SPI 功能被禁止，SPI 脚用作普通 IO 口
;DORD：SPI 数据传输顺序。
;      1：先传低位 LSB
;      0：先传高位 MSB
;MSTR：主 / 从模式选择位
;CPOL：SPI 时钟信号极性选择位
;      1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
;      0：SPI 空闲时 SPICLK = 0，前跳变沿是上升沿，后跳变沿是下降沿。
;CPHA：SPI 时钟信号相位选择位
;      1：数据由 SPICLK 前跳变沿驱动到口线，后跳变沿采样。
;      0：当 SS 脚为低(SSIG=0)时数据被驱动到口线，并且在 SPICLK 后跳变沿数据
;         被改变(被驱动到口线)，在 SPICLK 前跳变沿数据被采样。注意：SSIG = 1
;         时操作未定义。
;SPR1-SPR0：主模式时 SPI 时钟速率选择
;      00：fosc/4
;      01：fosc/16
;      10：fosc/64
```

```
;       11：fosc/128
;
;     当 CPHA=0，SSIG 必须等于零并且在传输时 SS 脚也必须一直保持为低。当 SS 有效
;(=0)时向 SPDATA 寄存器写数据就会发生写冲突错误，WCOL 标志被置 1。
;     当 CPHA=1，SSIG 可以等于 0 或 1。如果 SSIG=0，SS 脚在连续的传输时为 0(可以
;一直保持为 0)。当系统中只有一个主和一个从 SPI 时，这是首选配置。
;--------------------
;SPI 状态寄存器
;          7      6      5      4      3      2      1      0
;SPSTAT  SPIF   WCOL     -      -      -      -      -      -
;SPIF：SPI 传输结束标志。当一次传输结束时， SPIF 被置 1，如果 SPI 中断被打开：
;       ESPI(AUXR.3)=1，EADC_SPI(IE.5)=1，EA(IE.7)=1，就引起中断。如果原来 SPI
;       由 SS 脚确定为是主模式(SSIG=0,SS=1)，当 SS 变成 0 时，SPIF 也会被置 1，
;       表示"模式改变"。向 SPIF 位写 1 将该标志清 0。
;WCOL：SPI 写冲突标志。当一个数据还在传输时，又向数据寄存器 SPDAT 写入数据，WCOL
;       就会被被置 1。向 WCOL 位写 1 将该标志清 0。
;--------------------
;SPI 主/从模式选择
;
;SPEN SSIG SS MSTR    模式    MISO   MOSI  SPICLK     注释
;  0    X   X    X    禁止 SPI   输入   输入   输入      禁止 SPI
;  1    0   0    0       从      输出   输入   输入      被选为从
;  1    0   1    0     未选从    输入   输入   输入      从，但没有被选中
;  1    0   0  1->0      从      输出   输入   输入      由主变为从
;  1    0   1    1       主      输入   输出   输出
;  1    1   X    0       从      输出   输入   输入      从
;  1    1   X    1       主      输入   输出   输出      主
```

12.4 SPI 功能测试程序 3(适用于互为主从系统)

```
;/* ---  STC International Limited ------------------------------------------ */
;/* ---  宏晶科技     姚永平     2006/9/5     V1.0 ------------------------------ */
;/* ---  one_master_more_slave ---------------------------------------------- */
;/* ---  STC12C5412AD, STC12C5410AD, STC12C5408AD --------------------------- */
;/* ---  STC12C5406AD, STC12C5404AD, STC12C5402AD --------------------------- */
;/* ---  STC12C5052AD, STC12C4052AD ----------------------------------------- */
;/* ---  STC12C2052AD, STC12C1052AD, STC12C0552AD --------------------------- */
;/* ---  Mobile: 13922805190 ------------------------------------------------ */
;/* ---  Fax: 0755-82944243 ------------------------------------------------- */
;/* ---  Tel: 0755-82948409 ------------------------------------------------- */
;/* ---  Web: www.mcu-memory.com -------------------------------------------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 ------ -*/
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序------- */
;-------------------------------------------------------------------------------
;1. 本示例程序演示 STC12C2052AD, STC12C5410AD 系列 MCU 的 SPI 功能, 适用于互为
;    主从系统。
;-------------------------------------------------------------------------------
;2. 硬件连接:
;
;                    单片机 #1                      单片机 #2
;                +--------------+               +--------------+
;                |         MISO |<<------------>>| MISO         |
;                |              |               |              |
;                |         MOSI |<<------------>>| MOSI         |
;     PC  <<-->>|RS232         |               |         RS232|<<-->>PC
;串口调试助手    |         SCLK |<<------------>>| SCLK         |  串口调试助手
;                |           __ |               | __           |
;                |           SS |<<------------>>| SS           |
;                |              |               |              |
;                +--------------+               +--------------+
;
;3. SPI 通讯过程：
;    双方初始化时将自己设置成忽略 SS 脚的 SPI 从模式。当一方要主动发送数据时,
;先检测 SS 脚的电平，如果 SS 脚是高电平，就将自己设置成忽略 SS 脚的主模式。
;    主单片机与从单片机的 SPI 8 位移位寄存器连接成一个循环的 16 位移位寄存器。
;当主单片机程序向 SPDAT 写入一个字节时，立即启动一个连续的 8 位移位通讯过程:
;主单片机的 SCLK 脚向从单片机的 SCLK 脚发出一串脉冲，主单片机 SPDAT 寄存器中
;的数据移到了从单片机的 SPDAT 寄存器中; 与此同时，从单片机 SPDAT 寄存器中的
;数据移到了主单片机的 SPDAT 寄存器中。利用这样的数据交换机制，既可向从单片机
;发送数据，又可读从单片机中的数据。
;
;4. 程序重点提示
;      通讯双方平时将 SPI 置成没有被选中的从模式。在该模式下，MISO、MOSI、SPICLK
; 均为输入，当多个 MCU 的 SPI 接口以此模式并联时不会发生总线冲突。这种特性在互
;  为主从、一主多从等应用中很有用。
```

```
;5. 注意事项
;  a) 互为主从模式时，双方的 SPI 速率必须相同，因此要使用外部晶体振荡器，双方
;     的晶体频率也要相同。
;  b) SPI 特殊功能寄存器的详细说明见本程序的后部，也可参阅本公司的《应用指南》。
;6. 使用方法
;  a) 汇编后的程序代码下载到两个单片机中。
;  b) 用 PC 机串口调试助手(STC 的 ISP 下载程序 STC-ISP.exe 3.2 以上版本提供了该
;     功能) 向单片机 RS232 通讯口发送一串数据。
;  c) 在串口调试助手接收区观察接收的数据。
;-------------------------------------------------------------------------------
#include "STC12C5410AD_SFR.ASM"

;定义常量
;-----------------
;定义波特率自动重装数常量
;以下波特率是 PCON.7 = 0 时的数值，若使 PCON.7 = 1 可将波特率加倍
;RELOAD_COUNT_LOW  EQU 0EEH   ;Fosc=33.000MHz, Baud = 4774, 1T 运行时 Baud=57600
;RELOAD_COUNT_LOW  EQU 0FFH   ;Fosc=22.1184MHz, Baud = 57600
RELOAD_COUNT_LOW  EQU 0FBH   ;Fosc=18.432MHz, Baud=9600, 1T 运行时 Baud=115200
;RELOAD_COUNT_LOW   EQU 0F6H   ;Fosc=18.432MHz, Baud=4800, 1T 运行时 Baud=57600
;RELOAD_COUNT_LOW  EQU 0ECH    ;Fosc=18.432MHz, Baud=2400, 1T 运行时 Baud=28800
;RELOAD_COUNT_LOW  EQU 0D8H    ;Fosc=18.432MHz, Baud=1200, 1T 运行时 Baud=14400
;RELOAD_COUNT_LOW  EQU 0FFH   ;Fosc=11.059MHz, Baud = 28800、
;-----------------
;定义常量
;SPI 控制寄存器
;          7      6      5      4      3      2      1      0
;SPCTL   SSIG   SPEN   DORD   MSTR   CPOL   CPHA   SPR1   SPR0

;定义 SPI 模式常量
CONFIG_MASTER       EQU  11010000B  ;0D4H, 忽略 SS 脚的主模式
CONFIG_SLAVE        EQU  01000000B  ;40H, SS 脚用于确定器件为主机还是从机。
                                    ;SS=1 时为未选从模式，没有总线冲突
                                    ;    MISO,MOSI,SPICLK= 输入,输入,输入
                                    ;SS=0 时为从模式，
                                    ;    MISO,MOSI,SPICLK= 输出,输入,输入

;CONFIG_MASTER       EQU  11010100B
;CONFIG_SLAVE        EQU  01000100B  ;40H, SS 脚用于确定器件为主机还是从机。

;CONFIG_MASTER       EQU  11011000B
;CONFIG_SLAVE        EQU  01001000B  ;40H, SS 脚用于确定器件为主机还是从机。

;CONFIG_MASTER       EQU  11011100B
;CONFIG_SLAVE        EQU  01001100B

;-----------------------------------------------
;定义 SPI 脚
```

```
SCLK            EQU P1.7
MISO            EQU P1.6
MOSI            EQU P1.5
SS              EQU P1.4
;-----------------------------------------------
;定义变量
Flags           EQU 20H
SPI_Receive     EQU Flags.0         ;收到数据标志
TimeOver        EQU Flags.1         ;RS232 接收下一个字节超时

Receive_buffer  EQU 21H             ;缓冲区指针
Receive_Number  EQU 22H             ;已接收数据长度

SPI_buffer      EQU 23H             ;保存 SPI 接口收到的数据
Time_Over       EQU 24H             ;超时计数
;-----------------------------------------------
    ORG   0000H
    AJMP  MAIN
;-----------------------------------------------
    ORG   000BH                     ;定时器 0 中断服务程序入口
    AJMP  timer0_Routine
;-----------------------------------------------
    ORG   002BH                     ;ADC_SPI 中断服务程序入口
    LJMP  ADC_SPI_Interrupt_Routine ;ADC_SPI 中断服务程序
;-----------------------------------------------
    ORG   0080H
MAIN:
    MOV   SP, #0E5H
    ACALL Init_System               ;系统初始化
;--------------------
Check_Receive:
    JB    RI, RS232_GetFirstByte    ;RS232 收到第一个字节
    JB    SPI_Receive, SPI_GetFirstByte  ;SPI 收到第一个字节
    SJMP  Check_Receive             ;循环

;-----------------------------------------------
;-----------------------------------------------
RS232_GetFirstByte:                 ;RS232 收到第一个字节
    MOV   Receive_buffer, #30H
    MOV   Receive_Number, #0B0H     ;最多接收 176 个字节，否则会破坏栈的内容
;--------------------
RS232_Received:                     ;RS232 收到一个字节
    MOV   R0, Receive_buffer        ;保存收到的字节
    MOV   @R0, SBUF
    CLR   RI
    DJNZ  Receive_Number, RS232_Receive_Next  ;判断接收缓冲区是否已满
    SJMP  SPI_Send                  ;接收缓冲区满，跳转
;--------------------
```

```
RS232_Receive_Next:                 ;准备接收下一个字节
    INC   Receive_buffer
    MOV   TH0, #0
    SETB  TR0                       ;启动 T0
    CLR   TimeOver
RS232_GetNextByteWait:               ;等待 RS232 传来的下一个字节
    JB    TimeOver, SPI_Send        ;超时，用 SPI 口发送 RS232 收到的字节
    JB    RI, RS232_Received
    SJMP  RS232_GetNextByteWait
;--------------------
SPI_Send:                           ;用 SPI 口发送 RS232 收到的字节
    MOV   A, #0B0H                  ;计算收到的字节数 -->R2
    CLR   C
    SUBB  A, Receive_Number
    MOV   R2, A
    MOV   R0, #30H
    ACALL ChangeToMarst             ;切换到忽略 SS 脚的主模式
    JC    ExitMarst                 ;切换失败，跳转
;--------------------
SPI_Send_Loop:
    MOV   A, @R0
    ACALL SPI_Send_Byte             ;将 A 中的数据发送到从 SPI
    INC   R0
    DJNZ  R2, SPI_Send_Loop
;--------------------
ExitMarst:
    CLR   TR0
    ACALL ChangeToSlave             ;SPI 接口切换到未选中的从模式
    SJMP  Check_Receive
;---------------------------------------------
;---------------------------------------------
SPI_GetFirstByte:                   ;SPI 收到第一个字节
    MOV   Receive_buffer, #30H
    MOV   Receive_Number, #0B0H     ;最多接收 176 个字节，否则会破坏栈的内容
;--------------------
SPI_Received:                       ;SPI 收到一个字节
    MOV   R0, Receive_buffer        ;保存收到的字节
    MOV   @R0, SPI_buffer
    CLR   SPI_Receive

    DJNZ  Receive_Number, SPI_Receive_Next  ;判断接收缓冲区是否已满
    SJMP  RS232_Send                ;接收缓冲区满，跳转
;--------------------
SPI_Receive_Next:                   ;准备接收下一个字节
    INC   Receive_buffer
    MOV   TH0, #0
    SETB  TR0                       ;启动 T0
    CLR   TimeOver
```

```
SPI_GetNextByteWait:                ;等待 SPI 传来的下一个字节
    JB    TimeOver, RS232_Send      ;超时，用 RS232 口发送 SPI 收到的字节
    JB    SPI_Receive, SPI_Received
    SJMP  SPI_GetNextByteWait
;--------------------
RS232_Send:                         ;用 RS232 口发送 SPI 收到的字节
    MOV   A, #0B0H                  ;计算收到的字节数 -->R2
    CLR   C
    SUBB  A, Receive_Number
    MOV   R2, A
    MOV   R0, #30H
;--------------------
RS232_Send_Loop:
    MOV   A, @R0
    ACALL RS232_Send_Byte           ;将 A 中的数据发送到 PC
    INC   R0
    DJNZ  R2, RS232_Send_Loop
;--------------------
    SJMP  Check_Receive
;---------------------------------------------
;---------------------------------------------
ChangeToMarst:                      ;切换到忽略 SS 脚的主模式
    SETB  SS                        ;如果 SS 脚未被拉低，就切换
    JB    SS, ChangeToMarst_a
    SETB  C                         ;切换到主模式失败
    RET

ChangeToMarst_a:
    CLR   IE.5                      ;禁止 AD 转换中断，SPI 中断
    MOV   SPCTL,#CONFIG_MASTER      ;忽略 SS 脚，设为主机。
    MOV   SPSTAT,#11000000B         ;清标志位 SPSTAT
    CLR   SS                        ;拉低 SS 脚，使对方成为从模式
    CLR   C                         ;本机切换到主模式成功
    RET
;---------------------------------------------
ChangeToSlave:                      ;SPI 接口切换到未选中的从模式
    SETB  SS                        ;释放 SS 脚
    MOV   SPCTL,#CONFIG_SLAVE
    MOV   SPSTAT,#11000000B         ;清标志位 SPSTAT
    SETB  IE.5                      ;允许 AD 转换中断，SPI 中断
    RET
;---------------------------------------------
Init_System:
    ACALL Initial_UART              ;初始化串口
    ACALL Initial_SPI               ;初始化 SPI

    MOV   Flags, #0                 ;清标志字
```

```
    SETB  ET0                       ;开 T0 中断
    SETB  EA                        ;开总中断

    RET
;-----------------------------------------------
Initial_UART:                       ;初始化串口
; SCON  Bit:   7       6     5     4     3     2     1     0
;           SM0/FE   SM1   SM2   REN   TB8   RB8    Tl    Rl
    MOV   SCON, #50H                ; 0101,0000 8 位可变波特率，无奇偶校验

    MOV   TMOD, #21H                ;T1 为自动重装模式
    MOV   TH1, #RELOAD_COUNT_LOW
    MOV   TL1, #RELOAD_COUNT_LOW
;    MOV   PCON, #80H                ;取消本行指令注释，波特率加倍。

    ;使以下两行有效，波特率快 12 倍，即波特率 = 4800*12 = 57600
    MOV   A, #01000000B             ;T1 以 1T 的速度计数，是普通 8051 的 12 倍
    ORL   AUXR, A

    SETB  TR1
    RET
;-----------------------------------------------
Initial_SPI:                        ;初始化 SPI 从模式
    MOV   SPCTL,#CONFIG_SLAVE
    MOV   SPSTAT,#11000000B         ;清 0 标志位 SPIF(SPSTAT.7), WCOL(SPSTAT.6)
                                    ;向该两个标志位写 "1" 会将它们清 0
    ORL   AUXR, #00001000B          ;令 ESPI(AUXR.3)=1,允许 SPIF(SPSTAT.7)产生中断
    SETB  EADC_SPI                  ;开 ADC 中断和 SPI 中断共享的总中断控制位
    RET
;-----------------------------------------------
RS232_Send_Byte:                    ;将 A 中的数据发送到 PC
    CLR   TI                        ;清零串口发送中断标志
    MOV   SBUF, A
    JNB   TI, $                     ;等待发送完毕，未发送完毕跳回本行
    CLR   TI                        ;清零串口发送中断标志
    RET
;-----------------------------------------------
SPI_Send_Byte:                      ;将 A 中的数据发送到从 SPI
    MOV   SPDAT, A
SPI_Send_Byte_Wait:
    MOV   A, SPSTAT
    ANL   A, #80H
    JZ SPI_Send_Byte_Wait           ;等待 SPIF=1 即等待 SPI 发送完毕
    MOV   SPSTAT,#11000000B         ;清标志位 SPSTAT，放弃发送时对方传来的字节
    RET
;-----------------------------------------------
timer0_Routine:                     ;定时器 0 中断服务程序
    SETB  TimeOver
```

```
    RETI
;-----------------------------------------------
ADC_SPI_Interrupt_Routine:          ;ADC_SPI 中断服务程序
;    PUSH  PSW
    ;SPI 中断服务程序
    MOV   SPI_buffer, SPDAT         ;保存收到的数据
    MOV   SPSTAT, #11000000B        ;0C0H，清 0 标志位 SPIF 和 WCOL
                ;特别注意：是向标志位 SPIF/WCOL 写 1，将 SPIF/WCOL 清成 0
                ;特别注意：不是向标志位 SPIF/WCOL 写 0，将 SPIF/WCOL 清成 0
    SETB  SPI_Receive               ;树立 SPI 端口收到数据标志
;    POP   PSW
    RETI
;-----------------------------------------------
    END
;-----------------------------------------------

;SS: 主 / 从器件选择。0= 从器件， 1= 主器件，其它条件参见 SPI 控制寄存器
;    SPCTL.7 SSIG。
;--------------------
;SPI 控制寄存器
;          7      6      5      4      3      2      1      0
;SPCTL   SSIG   SPEN   DORD   MSTR   CPOL   CPHA   SPR1   SPR0
;
;SSIG: 忽略 SS 脚，如果 SSIG=1，由 MSTR 位决定主模式或从模式，
;      如果 SSIG=0，由 SS 脚决定主模式或从模式。
;SPEN：SPI 使能位。如果 SPEN=0，SPI 功能被禁止，SPI 脚用作普通 IO 口
;DORD：SPI 数据传输顺序。
;      1：先传低位 LSB
;      0：先传高位 MSB
;MSTR：主 / 从模式选择位
;CPOL：SPI 时钟信号极性选择位
;      1：SPI 空闲时 SPICLK = 1，引导跳变沿是下降沿，后续跳变沿是上升沿。
;      0：SPI 空闲时 SPICLK = 0，引导跳变沿是上升沿，后续跳变沿是下降沿。
;CPHA：SPI 时钟信号相位选择位
;      1：数据由 SPICLK 引导跳变沿驱动到口线，后续跳变沿采样。
;      0：当 SS 脚为低(SSIG=0)时数据被驱动到口线，并且在 SPICLK 后续跳变沿数据
;         被改变(被驱动到口线)，在 SPICLK 引导跳变沿数据被采样。
;      注意：如果 SSIG=1，CPHA 就不能等于 1，否则操作不明确。
;SPR1-SPR0：主模式时 SPI 时钟速率选择
;      00：fosc/4
;      01：fosc/16
;      10：fosc/64
;      11：fosc/128
;
;    当 CPHA=0，SSIG 必须等于零并且在传输时 SS 脚也必须一直保持为低。当 SS 有效
;(=0)时向 SPDATA 寄存器写数据就会发生写冲突错误，WCOL 标志被置 1。
;    当 CPHA=1，SSIG 可以等于 0 或 1。如果 SSIG=0，SS 脚在连续的传输时为 0(可以
;一直保持为 0)。当系统中只有一个主和一个从 SIP 时，这是首选配置。
```

```
;--------------------
;SPI 状态寄存器
;           7      6      5      4      3      2      1      0
;SPSTAT   SPIF   WCOL     -      -      -      -      -      -
;SPIF：SPI 传输结束标志。当一次传输结束时， SPIF 被置 1，如果 SPI 中断被打开：
;       ESPI(AUXR.3)=1，EADC_SPI(IE.5)=1，EA(IE.7)=1，就引起中断。如果原来 SPI
;       由 SS 脚确定为是主模式(SSIG=0,SS=1)，当 SS 变成 0 时，SPIF 也会被置 1，
;       表示"模式改变"。向 SPIF 位写 1 将该标志清 0。
;WCOL：SPI 写冲突标志。当一个数据还在传输时，又向数据寄存器 SPDAT 写入数据，WCOL
;       就会被被置 1。向 WCOL 位写 1 将该标志清 0。
;--------------------
;SPI 主/从模式选择
;

;2006-12-28 测试结果：
;SPEN SSIG SS MATR   模式    MISO  MOSI SPICLK    注释
;  0    X   *   X   禁止 SPI   *      *     *      禁止 SPI 功能
;  1    0   0   X      从     输出   输入   输入     被选为从
;  1    0   1   0    未选从   输入   输入   输入     从，但没有被选中。当多个 MCU
                                                   ;的 SPI 接口以此模式并联时不会发生
                                                     ;总线冲突，这种特性在互为主从、一主
                                                     ;多从等应用时很有用。
;  1    0   1   1      主     输入   输出   输出     主
;  1    1   X   0      从     输出   输入   输入     从，MISO 电平随着 SPDAT 改变
;  1    1   X   1      主     输入   输出   输出     主，MOSI 电平随着 SPDAT 改变

;注：1．以上各模式中标注为输出的管脚均为推挽输出。
;    2．以上各模式中 SS 均是输入脚。
```

12.5 STC89 系列单片机和 STC12 系列单片机双 CPU 通信

--- 主机用软件实现 SPI 通信，从机用硬件实现 SPI 通信

```
;/* ---  STC International Limited ------------------------------------------ */
;/* ---  宏晶科技    姚永平    2006/7/1    V1.0 ---------------------------------- */
;/* ---  one_master_one_slave ----------------------------------------------- */
;/* ---  STC89C51RC,   STC89C52RC,   STC89C53RC    做主机用普通 I/O 口软件实现 SPI -- */
;/* ---  STC89C54RD+,  STC89C58RD+,  STC89C516RD+  做主机用普通 I/O 口软件实现 SPI -- */
;/* ---  STC12C5412AD, STC12C5410AD, STC12C5408AD  做从机用自身的硬件 SPI --------- */
;/* ---  STC12C5052AD, STC12C4052AD, STC12C2052AD  做从机用自身的硬件 SPI --------- */
;/* ---  Mobile: 13922805190 ------------------------------------------------ */
;/* ---  Fax: 0755-82944243 ------------------------------------------------- */
;/* ---  Tel: 0755-82948409 ------------------------------------------------- */
;/* ---  Web: www.MCU-Memory.com -------------------------------------------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序 ----------- */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序 ----------- */

;-------------------------------------------------------------------------------
;1. 本示例程序演示 STC89C/LE5xx 系列 MCU 用任意 I/O 口实现 SPI 功能。本程序适用
;   于单主单从系统。从机的程序可参见 STC12 系列单片机单主单从 SPI 通信从机部分。
;-------------------------------------------------------------------------------
;2. 硬件连接:三线连接
;
;          单片机                                    从 SPI 器件
;
;      +--------------+  MISO    <-- 位流方向     MISO +--------------+
;      |   任意 I/O 口|<<----------------------------|     SPI      |
;      |              |                             |8 位移位寄存器 |
;      |   任意 I/O 口|---------------------------->>|              |
;      +-------+------+  MOSI    位流方向 -->     MOSI +-------^------+
;              |                                          |
;              |         SCLK                    SCLK     |
;              +--------------------->>-------------------+
;
;    除此之外，单片机的 RS-232 串行口通过 RS-232 转换器与 PC 机的 RS-232
;串行口相连接。
;-------------------------------------------------------------------------------
;3. SPI 通讯过程：
;3.1 通讯原理:
;    单片机每向"从 SPI 器件"(或从单片机)的 SCLK 脚发送一个脉冲，就有 1 bit
;单片机数据由 MOSI 脚传送到"从 SPI 器件"的 8 位移位寄存器中。与此同时有 1 bit
;"从 SPI 器件"数据由 MISO 脚传送到单片机中。也可以认为单片机和"从 SPI 器件"
;互换了 1 bit 数据。
;3.2 单片机发送 1 字节到"从 SPI 器件"
;    单片机的发送子程序每次将 8 bit 数据发送(由 8 个 SCLK 脉冲驱动) 到从
;SPI 器件的 8 位移位寄存器中。与此同时有 8 bit 数据由"从 SPI 器件"传回单片机,
;但单片机的发送子程序未予处理这 8 bit 数据。
```

```
;3.3 单片机接收"从 SPI 器件"的 1 字节
;     单片机的接收子程序每次向 SCLK 脚发送 8 个脉冲，将"从 SPI 器件"传回的
; 8位移位寄存器数据保存起来备用。与此同时有 8 bit 未知数据由单片机传送到了
;"从 SPI 器件"的 8位移位寄存器中，在实际应用中请注意对此进行处理。
;4.  SPI 工作模式
;     由三个特性决定 SPI 工作模式
;    a) 数据传输顺序：先传低位 LSB 还是先传高位 MSB，在本例中由 SOFT_DORD 设定
;    b) 时钟信号极性：空闲时 SCLK 脚的电平，即 SCLK 的"前跳变沿"是下降沿还是
;                      上升沿。在本例中由 SOFT_CPOL 设定时钟信号极性。
;    c) 时钟信号相位：第一个跳变沿发生在第一个 bit 传输周期的 0 度 还是 180 度。
;                     即数据由  SCLK 的"前跳变沿"还是"后跳变沿"驱动到 SPI
;                      口线。在本例中由 SOFT_CPHA 设定时钟信号相位。
;5. 使用方法
;    a) 修改程序，使 SPI 工作模式与 "从 SPI 器件" 的工作模式相同。
;    b) 将汇编后的程序代码下载到从单片机中。
;    c) 给电路板上电。
;    d) 用串口调试助手(STC 的 ISP 下载程序 STC-ISP.exe 3.2 以上版本提供了该功能)
;       向单片机发送一串数据。
;          单片机的 RS-232 串口每收到一个字节就立刻将收到的字节通过 SPI 口
;       发送到 "从 SPI 器件" 中。接下来接收 "从 SPI 器件" 的一个字节，并把这个
;       字节通过 RS-232 口发送到 PC 机显示出来。
;6. 从 SPI 器件
;    可以用 STC12C2052AD 和 STC12C5410AD 系列 MCU 作为 "从 SPI 器件" 调试本程序。
;7. 注意事项:
;       "从 SPI 器件" 的速度较慢时，要在 SPI 口接收，发送子程序中插入空操作指令
;    NOP。
;8. 实验结果：使用 STC12C5410AD，STC89C5xx 系列 MCU，SPI 口传输数据无误。
;--------------------------------------------------------------------------------
;定义常量
;-----------------
;定义波特率自动重装数常量
;以下波特率是 PCON.7 = 0 时的数值，若使 PCON.7 = 1 可将波特率加倍
RELOAD_8BIT_DATA  EQU 0FFH    ;Fosc=22.1184MHz, Baud = 57600
;RELOAD_8BIT_DATA  EQU 0FBH   ;Fosc=18.432MHz, Baud=9600, 1T 运行时 Baud=115200
;RELOAD_8BIT_DATA   EQU 0F6H  ;Fosc=18.432MHz, Baud=4800, 1T 运行时 Baud=57600
;RELOAD_8BIT_DATA  EQU 0F3H   ;Fosc=12.0000, Baud = 2403
;RELOAD_8BIT_DATA  EQU 0FFH   ;Fosc=11.059MHz, Baud = 28800
;--------------------------------------------------------------------------------
;定义特殊功能寄存器
AUXR    EQU 8EH
;--------------------------------------------------------------------------------
;定义 SPI 脚
;SCLK    EQU P1.7
;MISO    EQU P1.6
;MOSI    EQU P1.5
SCLK    EQU P1.0
MISO    EQU P1.1
MOSI    EQU P1.2
```

```
;-------------------------------------------------------------------------------
;定义单片机管脚
LED_MCU_START        EQU P3.4
;-------------------------------------------------------------------------------
;定义 SPI 模式:
SOFT_DORD   EQU     1   ;SPI 数据传输顺序，1:先传低位 LSB，0:先传高位 MSB
SOFT_CPOL   EQU     1   ;SPI 时钟信号极性选择位
         ;1：SPI 空闲时 SPICLK = 1，前跳变沿是下降沿，后跳变沿是上升沿。
         ;0：SPI 空闲时 SPICLK = 0，前跳变沿是上升沿，后跳变沿是下降沿。
SOFT_CPHA   EQU     1   ;SPI 时钟信号相位选择位
         ;1：数据由 SPICLK 前跳变沿驱动到 SPI 口线，SPI 模块在后跳变沿采样数据。
         ;0：SPICLK 后跳变沿数据被改变(被驱动到口线)，在 SPICLK 前跳变沿数据被采样。
;-------------------------------------------------------------------------------
SET_SCLK_IDEL_VAL   MACRO           ;设置 SPI clock 时钟线空闲时电平
    if SOFT_CPOL
        SETB  SCLK                  ;1:SPI 空闲时 SPICLK=1,前跳变沿是下降沿,后跳变沿是上升沿
    else
        CLR   SCLK                  ;0:SPI 空闲时 SPICLK=0,前跳变沿是上升沿,后跳变沿是下降沿
    endif
ENDM

SET_SCLK_FOREPART_VAL  MACRO        ;设置 SCLK 脚 1 bit 周期前半程电平
    if SOFT_CPHA = 0
        if SOFT_CPOL = 0
            CLR   SCLK              ;SOFT_CPHA, SOFT_CPOL = 0,0
        else
            SETB  SCLK              ;SOFT_CPHA, SOFT_CPOL = 0,1
        endif
    else
        if SOFT_CPOL = 0
            SETB  SCLK              ;SOFT_CPHA, SOFT_CPOL = 1,0
        else
            CLR   SCLK              ;SOFT_CPHA, SOFT_CPOL = 1,1
        endif
    endif
ENDM

SET_SCLK_SECOND_HALF_VAL   MACRO    ;设置 SCLK 脚 1 bit 周期后半程电平
    if SOFT_CPHA = 0
        if SOFT_CPOL = 0
            SETB  SCLK              ;SOFT_CPHA, SOFT_CPOL = 0,0
        else
            CLR   SCLK              ;SOFT_CPHA, SOFT_CPOL = 0,1
        endif
    else
        if SOFT_CPOL = 0
            CLR   SCLK              ;SOFT_CPHA, SOFT_CPOL = 1,0
```

```
        else
            SETB  SCLK              ;SOFT_CPHA, SOFT_CPOL = 1,1
        endif
    endif
ENDM

OUTPUT_BIT_SPI_MOSI   MACRO         ;输出一 bit 到 MOSI 口线
    if SOFT_DORD
        RRC  A                      ;1:先传低位 LSB
    else
        RLC  A                      ;0:先传高位 MSB
    endif
    MOV   MOSI, C
ENDM

INPUT_BIT_SPI_MISO    MACRO         ;从 MISO 口线输入 1 bit
    MOV   C, MISO
    if SOFT_DORD
        RRC  A                      ;1:先传低位 LSB
    else
        RLC  A                      ;0:先传高位 MSB
    endif
ENDM
;-------------------------------------------------------------------------
;定义变量
SPI_buffer       EQU     30H        ;SPI 字节收发缓冲区
;-------------------------------------------------------------------------
    ORG   0000H
    LJMP  MAIN
;-------------------------------------------------------------------------
    ORG   0080H
MAIN:
    CLR   LED_MCU_START             ;点亮 MCU 开始工作指示灯
    MOV   SP, #7FH
    ACALL Init_System               ;系统初始化

Check_RS232:
    JNB   RI, Check_RS232           ;判 RS-232 串口中收到数据否
    ;单片机 RS-232 串口已收到新的数据
    ACALL Get_Byte_From_RS232       ;单片机将 RS-232 串口中收到的数据送到累加器 A

;   ACALL RS232_Send_Byte           ;调试用，将累加器 A 中的数据发送到 PC 机

    ACALL SPI_Send_Byte             ;单片机将累加器 A 中的数据发送到"从 SPI 器件"
    ACALL Delay                     ;延时，等待"从 SPI 器件"准备好发送数据
    ACALL SPI_Receive_Byte          ;接收"从 SPI 器件"一个字节
    ;单片机 SPI 端口已收到新的数据
```

```
    MOV   A, SPI_buffer             ;将 "从 SPI 器件" 发回的数据"送到累加器 A
    ACALL RS232_Send_Byte           ;将累加器 A 中的数据发送到 PC 机
    SJMP  Check_RS232
;-------------------------------------------------------------------------
Init_System:
    ACALL Initial_UART              ;初始化串口
    ACALL Initial_SPI               ;初始化 SPI
    SETB  EA                        ;开总中断
    RET
;-------------------------------------------------------------------------
Initial_UART:                       ;初始化串口
; SCON  Bit:   7       6     5     4     3     2     1     0
;           SM0/FE   SM1   SM2   REN   TB8   RB8    TI    RI
    MOV   SCON, #50H                ;0101,0000 8 位可变波特率，无奇偶校验

    MOV   TMOD, #21H                ;T1 为自动重装模式
    MOV   TH1, #RELOAD_8BIT_DATA
    MOV   TL1, #RELOAD_8BIT_DATA

;    MOV   PCON, #80H                ;取消本行指令注释，波特率加倍。

    ;使以下两行有效，波特率快 12 倍
    MOV   A, #01000000B             ;T1 以 1T 的速度计数，是普通 8051 的 12 倍
    ORL   AUXR, A

    SETB  TR1                       ;启动定时器 1 开始计数
    RET
;-------------------------------------------------------------------------
Initial_SPI:                        ;初始化 SPI
    SET_SCLK_IDEL_VAL               ;设置 SPI clock 时钟线空闲时电平
    SETB  MISO
    SETB  MOSI
    RET
;-------------------------------------------------------------------------
RS232_Send_Byte:                    ;RS232 串口发送一个字节
    CLR   TI                        ;清零串口发送中断标志
    MOV   SBUF, A
RS232_Send_Wait:
    JNB   TI, RS232_Send_Wait       ;等待发送完毕，未发送完毕跳回本行
    CLR   TI                        ;清零串口发送中断标志
    RET
;-------------------------------------------------------------------------
Get_Byte_From_RS232:                ;取 RS-232 串口中收到的数据送累加器 A
    MOV   A, SBUF
    CLR   RI
    RET
```

```
;-------------------------------------------------------------------------------
;将 A 中的数据用 SPI 口发送出去
SPI_Send_Byte:                      ;SPI 发送一个字节
    SET_SCLK_IDEL_VAL               ;设置 SPI clock 时钟线空闲时电平
    MOV   R2, #8
SPI_Send_Loop:
    OUTPUT_BIT_SPI_MOSI             ;输出一 bit 到 MOSI 口线
    SET_SCLK_FOREPART_VAL           ;设置 SCLK 脚 1 bit 周期前半程电平
;   NOP                             ;为降低 SPI 通讯速率可插入数个 NOP
;   NOP
;   NOP
    SET_SCLK_SECOND_HALF_VAL        ;设置 SCLK 脚 1 bit 周期后半程电平
;   NOP                             ;为降低 SPI 通讯速率可插入数个 NOP
;   NOP
    DJNZ  R2, SPI_Send_Loop
    SET_SCLK_IDEL_VAL               ;设置 SPI clock 时钟线空闲时电平
    RET
;-------------------------------------------------------------------------------
;用 SPI 接收一个字节送 A 和 SPI_buffer 中
SPI_Receive_Byte:                   ;SPI 接收一个字节
    SET_SCLK_IDEL_VAL               ;设置 SPI clock 时钟线空闲时电平
    MOV   R2, #8
SPI_Receive_Loop:
    SET_SCLK_FOREPART_VAL           ;设置 SCLK 脚 1 bit 周期前半程电平
    NOP                             ;等待 MISO 口线数据稳定，这个 NOP 是必需的
    NOP                             ;对于高速的 1T 单片机，这个 NOP 是必需的
;   NOP                             ;为降低 SPI 通讯速率可插入数个 NOP
    INPUT_BIT_SPI_MISO              ;从 MISO 口线输入 1 bit
    SET_SCLK_SECOND_HALF_VAL        ;设置 SCLK 脚 1 bit 周期后半程电平
;   NOP                             ;为降低 SPI 通讯速率可插入数个 NOP
;   NOP
    DJNZ  R2, SPI_Receive_Loop
    SET_SCLK_IDEL_VAL               ;设置 SPI clock 时钟线空闲时电平
    MOV   SPI_buffer, A
    RET
;-------------------------------------------------------------------------------
Delay:
    MOV   R2, #50H
Delay_Loop:
    DJNZ  R2, Delay_Loop
    RET
;-------------------------------------------------------------------------------
    END
;-------------------------------------------------------------------------------
```

附录 A： 内部扩展数据 RAM 的使用

内部数据 RAM 存储器

STC12C5410AD 系列单片机内部有 256 字节常规的 RAM,256 字节的扩展 RAM

器件的内部常规数据存储器由 3 部分组成：

1.低 128 字节 RAM（00H～7FH），可直接和间接寻址,用“MOV”和“MOV @Ri”

2.高 128 字节 RAM（80H～FFH），间接寻址，用“MOV @Ri”

3.特殊功能寄存器（80H～FFH），只可直接寻址， 用“MOV”

由于高 128 字节 RAM 和 SFR（特殊功能寄存器）占用相同的地址，因此高 128 字节 RAM 空间必须用间接寻址（MOV @Ri）来区分。特殊功能寄存器（80H～FFH），只可直接寻址（用“MOV”）来区分。尽管 RAM 和 SFR 的地址相同，但它们在物理上是独立的。

扩展数据 RAM

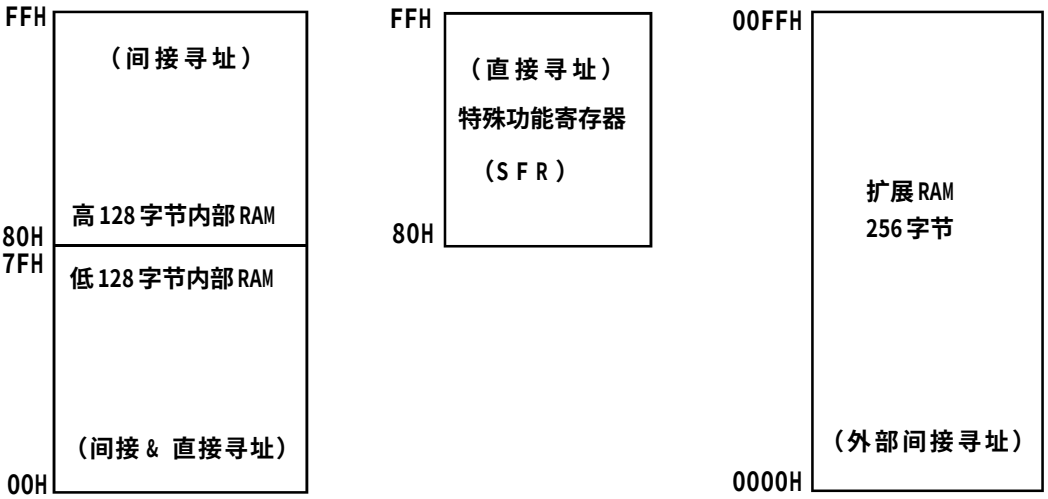
STC12C5410AD 系列有 256 字节的扩展 RAM，称其为 XRAM（附加 RAM），用“MOVX”寻址。

扩展的 256 字节 RAM（0000H～00FFH），通过 MOVX 指令间接寻址。

使用“MOVX @DPTR” / “MOVX @Ri”

C 语言中,可使用 xdata 声明存储类型即可,如:

```
unsigned char xdata  i = 0;
```



附录 B： 内部常规 256 字节 RAM 间接寻址测试程序

```
;/* ---  STC International Limited ------------------------------------- */
;/* ---  宏晶科技     姚永平     2006/1/6     V1.0 ---------------------------- */
;/* ---  STC12C5410AD 系列单片机 内部常规 RAM 间接寻址测试程序 ---------------- */
;/* ---  Mobile: 13922805190 ------------------------------------------- */
;/* ---  Fax: 0755-82944243 -------------------------------------------- */
;/* ---  Tel: 0755-82948409 -------------------------------------------- */
;/* ---  Web: www.mcu-memory.com --------------------------------------- */
;/* ---  本演示程序在 STC-ISP Ver 3.0A.PCB 的下载编程工具上测试通过 ----------- */
;/* ---  如果要在程序中使用该程序,请在程序中注明使用了宏晶科技的资料及程序  ---- */
;/* ---  如果要在文章中引用该程序,请在文章中注明使用了宏晶科技的资料及程序  ---- */

TEST_CONST EQU  5AH
;TEST_RAM  EQU  03H
     ORG   0000H
     LJMP  INITIAL

     ORG   0050H
INITIAL:
     MOV   R0,  #253

     MOV   R1,  #3H
TEST_ALL_RAM:
     MOV   R2,  #0FFH
TEST_ONE_RAM:
     MOV   A,   R2
     MOV   @R1, A
     CLR   A
     MOV   A,   @R1

     CJNE  A,   2H,  ERROR_DISPLAY
     DJNZ  R2,  TEST_ONE_RAM
     INC   R1
     DJNZ  R0,  TEST_ALL_RAM

OK_DISPLAY:
     MOV   P1,  #11111110B
Wait1:
     SJMP  Wait1

ERROR_DISPLAY:
     MOV   A,   R1
     MOV   P1,  A
Wait2:
     SJMP  Wait2
     END
```

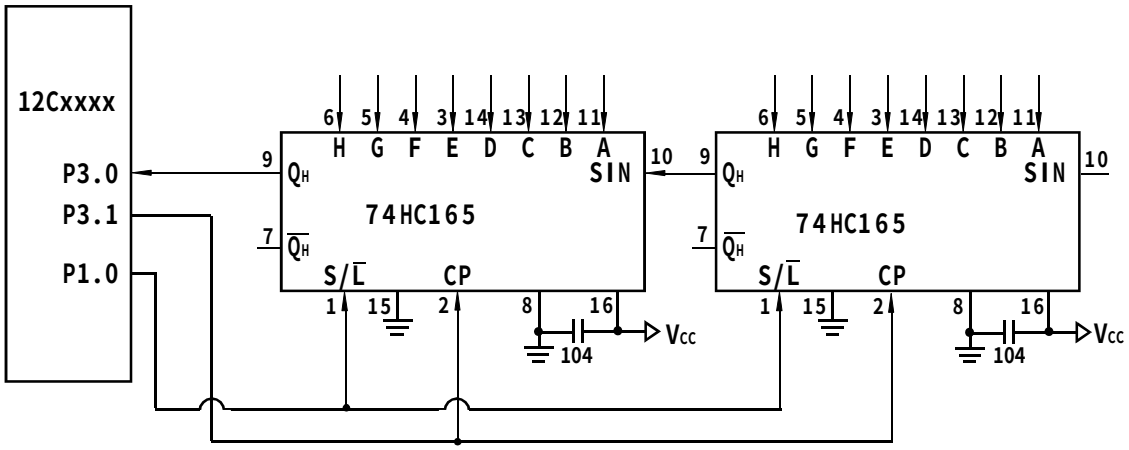
附录 C： 用串行口扩展 I/O 接口

STC12C5410 串行口的方式 0 可用于 I/O 扩展。如果在应用系统中，串行口未被占用，那么将它用来扩展并行 I/O 口是一种经济、实用的方法。

在操作方式 0 时，串行口作同步移位寄存器，其波特率是固定的，为 fosc/12（fosc 为振荡器频率）。数据由 RXD 端（P3.0）出入，同步移位时钟由 TXD 端（P3.1）输出。发送、接收的是 8 位数据，低位在先。

一、用 74HC165 扩展并行输入口

下图是利用两片 74HC165 扩展二个 8 位并行输入口的接口电路图。



74HC165 是 8 位并行置入移位寄存器。当移位 / 置入端(S/$\overline{L}$)由高到低跳变时，并行输入端的数据置入寄存器；当 S/$\overline{L}$=1，且时钟禁止端（第 15 脚）为低电平时，允许时钟输入，这时在时钟脉冲的作用下，数据将由 Q_A 到 Q_H 方向移位。

上图中，TXD(P3.1)作为移位脉冲输出端与所有 74HC165 的移位脉冲输入端 CP 相连；RXD（P3.0）作为串行输入端与 74HC165 的串行输出端 Q_H 相连；P1.0 用来控制 74HC165 的移位与置入而同 S/$\overline{L}$ 相连；74HC165 的时钟禁止端（15 脚）接地，表示允许时钟输入。当扩展多个 8 位输入口时，两芯片的首尾（Q_H 与 S_{IN}）相连。

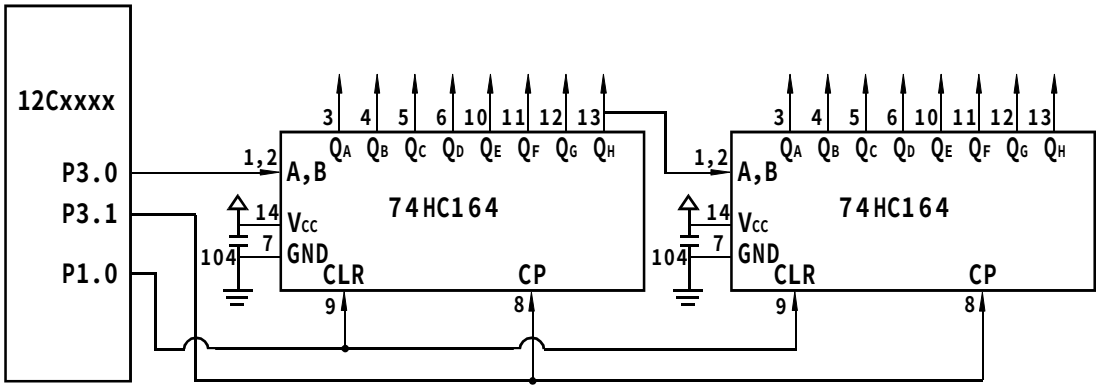
下面的程序是从16位扩展口读入5组数据（每组二个字节），并把它们转存到内部RAM 20H开始的单元中。

```
          MOV   R7，#05H          ；设置读入组数
          MOV   R0，#20H          ；设置内部 RAM 数据区首址
START：   CLR   P1.0              ；并行置入数据，S/L=0
          SETB  P1.0              ；允许串行移位 S/L=1
          MOV   R1，#02H          ；设置每组字节数，即外扩 74LS165 的个数
RXDATA：  MOV   SCON，#00010000B  ；设串行方式 0，允许接收，启动接收过程
WAIT：    JNB   RI，WAIT          ；未接收完一帧，循环等待
          CLR   RI                ；清 RI 标志，准备下次接收
          MOV   A，SBUF           ；读入数据
          MOV   @R0，A            ；送至 RAM 缓冲区
          INC   R0                ；指向下一个地址
          DJNZ  R1，RXDATA        ；为读完一组数据，继续
          DJNZ  R7，START         ；5 组数据未读完重新并行置入
          ……                      ；对数据进行处理
```

上面的程序对串行接收过程采用的是查询等待的控制方式，如有必要，也可改用中断方式。从理论上讲，按上图方法扩展的输入口几乎是无限的，但扩展的越多，口的操作速度也就越慢。

二、用 74HC164 扩展并行输出口

74HC164 是 8 位串入并出移位寄存器。下图是利用 74HC164 扩展二个 8 位输出口的接口电路。

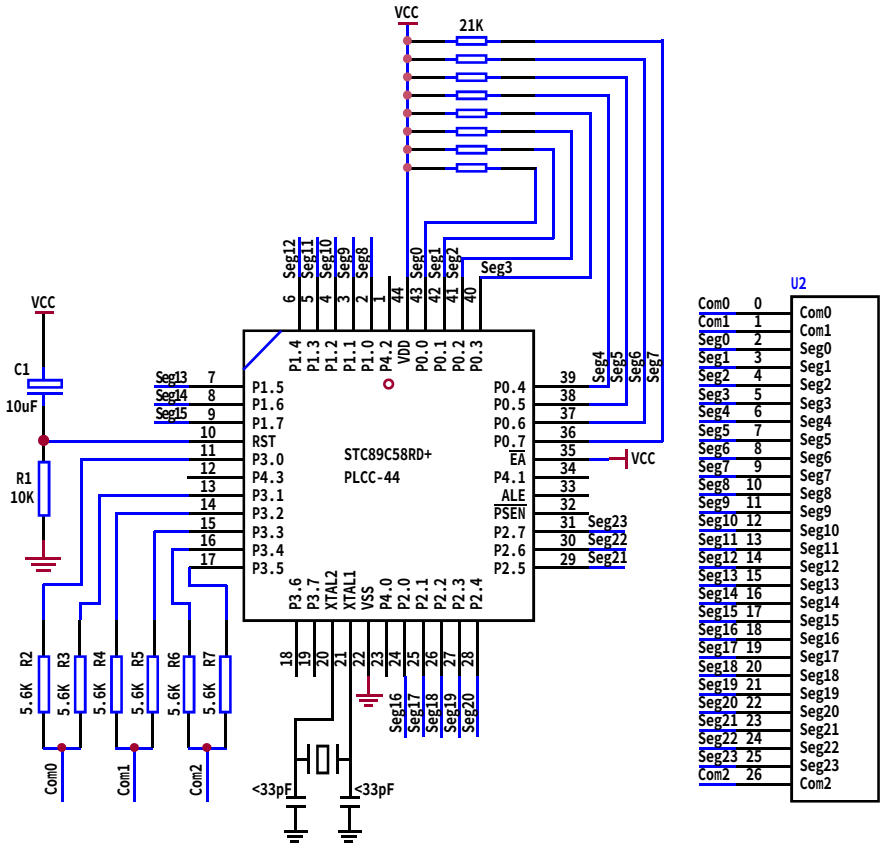


当单片机串行口工作在方式 0 的发送状态时，串行数据由 P3.0（RXD）送出，移位时钟由 P3.1（TXD）送出。在移位时钟的作用下，串行口发送缓冲器的数据一位一位地移入74HC164中。需要指出的是，由于74HC164无并行输出控制端，因而在串行输入过程中，其输出端的状态会不断变化，故在某些应用场合，在74HC164的输出端应加接输出三态门控制，以便保证串行输入结束后再输出数据。

下面是将 RAM 缓冲区 30H、31H 的内容串行口由 74HC164 并行输出的子程序。

```
START：  MOV     R7，#02H          ；设置要发送的字节个数
         MOV     R0，#30H          ；设置地址指针
         MOV     SCON，#00H        ；设置串行口方式 0
SEND：   MOV     A，@R0
         MOV     SBUF，A           ；启动串行口发送过程
WAIT：   JNB     TI，WAIT          ；一帧数据未发送完，循环等待
         CLR     TI
         INC     R0                ；取下一个数
         DJNZ    R7，SEND
         RET
```

附录D: STC 单片机普通 I/O 口驱动 LCD 显示



本资料不提供技术支持，请自行消化吸收

```
NAME LcdDriver
$include(STC89C51RC.h)

;*******************************************************************************
;the LCD is 1/3 duty and 1/3 bias; 3Com*24Seg; 9 display RAM;
;
;                    Bit7    Bit6   Bit5   Bit4   Bit3   Bit2   Bit1   Bit0
;Com0:  Com0Data0:   Seg7    Seg6   Seg5   Seg4   Seg3   Seg2     Seg1   Seg0
;       Com0Data1:   Seg15   Seg14  Seg13  Seg12  Seg11  Seg10    Seg9   Seg8
;       Com0Data2:   Seg23   Seg22  Seg21  Seg20  Seg19  Seg18    Seg17  Seg16
;Com1:  Com1Data0:   Seg7    Seg6   Seg5   Seg4   Seg3   Seg2     Seg1   Seg0
;       Com1Data1:   Seg15   Seg14  Seg13  Seg12  Seg11  Seg10    Seg9   Seg8
;       Com1Data2:   Seg23   Seg22  Seg21  Seg20  Seg19  Seg18    Seg17  Seg16
;Com2:  Com2Data0:   Seg7    Seg6   Seg5   Seg4   Seg3   Seg2     Seg1   Seg0
;       Com2Data1:   Seg15   Seg14  Seg13  Seg12  Seg11  Seg10    Seg9   Seg8
;       Com2Data2:   Seg23   Seg22  Seg21  Seg20  Seg19  Seg18    Seg17  Seg16
;*******************************************************************************
;Com0:  P3^0,P3^1   when P3^0 = P3^1 = 1       then Com0=VCC(=5V);
;                        P3^0 = P3^1 = 0       then Com0=GND(=0V);
;                        P3^0 = 1, P3^1=0      then Com0=1/2 VCC;
;Com1:  P3^2,P3^3   the same as the Com0
;Com2:  P3^4,P3^5   the same as the Com0
;

sbit SEG0  =P0^0
sbit SEG1  =P0^1
sbit SEG2  =P0^2
sbit SEG3  =P0^3
sbit SEG4  =P0^4
sbit SEG5  =P0^5
sbit SEG6  =P0^6
sbit SEG7  =P0^7
sbit SEG8  =P1^0
sbit SEG9  =P1^1
sbit SEG10 =P1^2
sbit SEG11 =P1^3
sbit SEG12 =P1^4
sbit SEG13 =P1^5
sbit SEG14 =P1^6
sbit SEG15 =P1^7
sbit SEG16 =P2^0
sbit SEG17 =P2^1
sbit SEG18 =P2^2
sbit SEG19 =P2^3
```

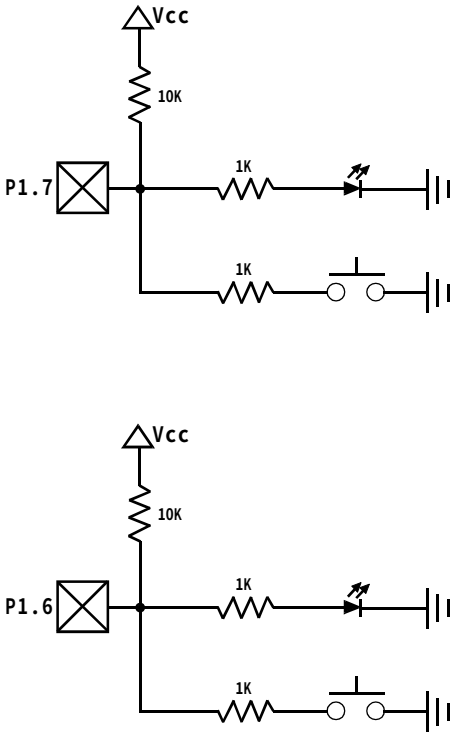
```
sbit SEG20 =P2^4
sbit SEG21 =P2^5
sbit SEG22 =P2^6
sbit SEG23 =P2^7
;*******************************************************************************************

;======Interrupt==============================
       CSEG AT 0000H
       LJMP start

       CSEG AT 000BH
       LJMP int_t0

;======register==============================
lcdd_bit SEGMENT BIT
       RSEG lcdd_bit
       OutFlag:       DBIT 1          ;the output display reverse flag

lcdd_data SEGMENT DATA
      RSEG lcdd_data
      Com0Data0:     DS   1
      Com0Data1:     DS   1
      Com0Data2:     DS   1
      Com1Data0:     DS   1
      Com1Data1:     DS   1
      Com1Data2:     DS   1
      Com2Data0:     DS   1
      Com2Data1:     DS   1
      Com2Data2:     DS   1
      TimeS:         DS   1

;======Interrupt Code=========================
t0_int SEGMENT CODE
     RSEG  t0_int
     USING 1
;*************************************************************
;Time0 interrupt
;ths system crystalloid is 22.1184MHz
;the time to get the Time0 interrupr is 2.5mS
;the whole duty is 2.5mS*6=15mS, including reverse
;*************************************************************
int_t0:
    ORL  TL0,#00H
    MOV  TH0,#0EEH
    PUSH ACC
    PUSH PSW
```

```
    MOV   PSW,#08H
    ACALL OutData
    POP   PSW
    POP   ACC
    RETI

;======SUB CODE================================
uart_sub SEGMENT CODE
        RSEG  uart_sub
        USING 0
;*********************************************************************
;initial the display RAM data
;if want to display other,then you may add other data to this RAM
;Com0:   Com0Data0,Com0Data1,Com0Data2
;Com1:   Com1Data0,Com1Data1,Com1Data2
;Com2:   Com2Data0,Com0Data1,Com0Data2
;*********************************************************************
InitComData:                      ;it will display "11111111"
    MOV Com0Data0,#24H
    MOV Com0Data1,#49H
    MOV Com0Data2,#92H
    MOV Com1Data0,#92H
    MOV Com1Data1,#24H
    MOV Com1Data2,#49H
    MOV Com2Data0,#00H
    MOV Com2Data1,#00H
    MOV Com2Data2,#00H
    RET

;*********************************************************************
;reverse the display data
;*********************************************************************
RetComData:
    MOV  R0,#Com0Data0              ;get the first data address
    MOV  R7,#9
RetCom_0:
    MOV  A,@R0
    CPL  A
    MOV  @R0,A
    INC  R0
    DJNZ R7,RetCom_0
    RET
```

```
;*********************************************************************
;get the display Data and send to Output register
;*********************************************************************
OutData:
    INC  TimeS
    MOV  A,TimeS
    MOV  P3,#11010101B              ;clear display,all Com are 1/2VCC and invalidate
    CJNE A,#01H,OutData_1           ;judge the duty
    MOV  P0,Com0Data0
    MOV  P1,Com0Data1
    MOV  P2,Com0Data2
    JNB  OutFlag,OutData_00
    MOV  P3,#11010111B              ;Com0 is work and is VCC
    RET
OutData_00:
    MOV  P3,#11010100B              ;Com0 is work and is GND
    RET
OutData_1:
    CJNE A,#02H,OutData_2
    MOV  P0,Com1Data0
    MOV  P1,Com1Data1
    MOV  P2,Com1Data2
    JNB  OutFlag,OutData_10
    MOV  P3,#11011101B              ;Com1 is work and is VCC
    RET
OutData_10:
    MOV  P3,#11010001B              ;Com1 is work and is GND
    RET
OutData_2:
    MOV  P0,Com2Data0
    MOV  P1,Com2Data1
    MOV  P2,Com2Data2
    JNB  OutFlag,OutData_20
    MOV  P3,#11110101B              ;Com2 is work and is VCC
    SJMP OutData_21
OutData_20:
    MOV  P3,#11000101B              ;Com2 is work and is GND
OutData_21:
    MOV   TimeS,#00H
    ACALL RetComData
    CPL   OutFlag
    RET
```

```
;======Main Code==============================
uart_main SEGMENT CODE
        RSEG  uart_main
        USING 0

start:
           MOV   SP,#40H
           CLR   OutFlag
           MOV   TimeS,#00H
           MOV   TL0,#00H
           MOV   TH0,#0EEH
           MOV   TMOD,#01H
           MOV   IE,#82H
           ACALL InitComData
           SETB  TR0
Main:
     NOP
     SJMP Main

END
```

附录E： 一个I/O口驱动发光二极管并扫描按键



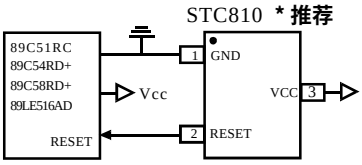
利用STC12系列单片机的I/O口可设置成弱上拉,强上拉(推挽)输出,仅为输入(高阻),开漏四种模式的特性,可以利用 STC12 系列单片机的 I/O 口同时作为发光二极管驱动及按键检测用，可以大幅节省 I/O 口。

当驱动发光二极管时，将该 I/O 口设置成强推挽输出，输出高即可点亮发光二极管。

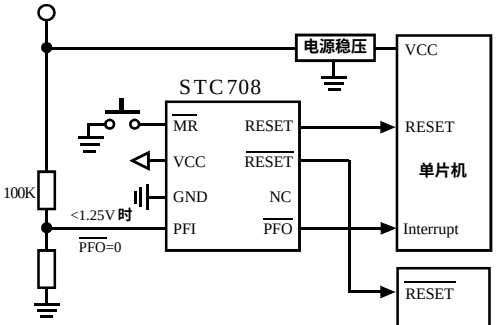
当检测按键时，将该 I/O 口设置成弱上拉输入，再读外部口的状态，即可检测按键。

附录F： 典型MCU/DSP/uC 复位、电源监控、外部看门狗专用电路

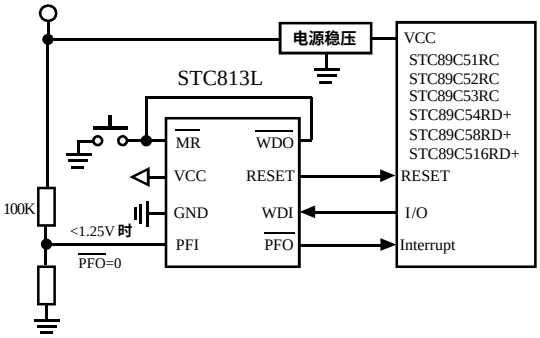
1.高电平复位信号输出



上电复位，掉电复位,人民币1元

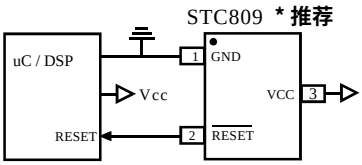


上电复位，掉电复位,外部手动复位，
电源稳压块前端掉电检测 ，
高 / 低电平 2 路复位信号输出

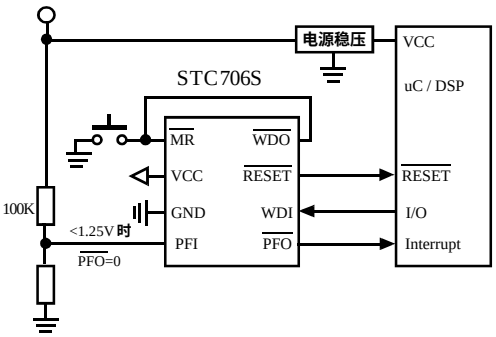


上电复位，掉电复位,外部手动复位，
电源稳压块前端掉电检测，
外部硬件看门狗

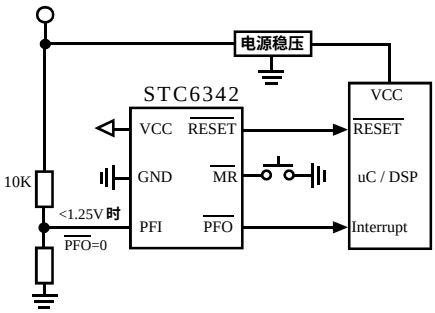
2.低电平复位信号输出



上电复位，掉电复位,人民币1元



上电复位，掉电复位,外部手动复位，
电源稳压块前端掉电检测，
外部硬件看门狗



上电复位，掉电复位,外部手动复位，稳压块前端掉电检测

使用外部专用复位电路的好处：

1.确保上电时，在用户设定的电源电压之上，时钟振荡稳定后，单片机才开始工作

2.确保掉电时，在用户设定的电源电压之下，立即让单片机复位，以免单片机误动作

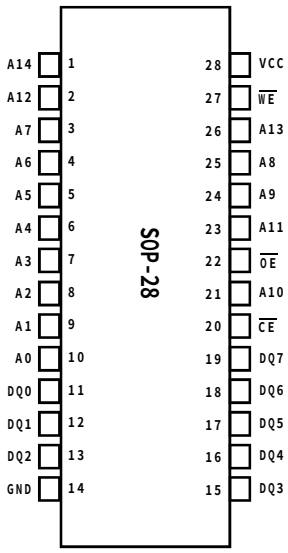
3.具有电源稳压块前端掉电检测的专用复位电路，确保掉电前有充分的时间保存数据

4.复位门槛电压可选： L:4.63V; M:4.38V; J: 4.00V; T:3.08V; S:2.93V; R:2.63V

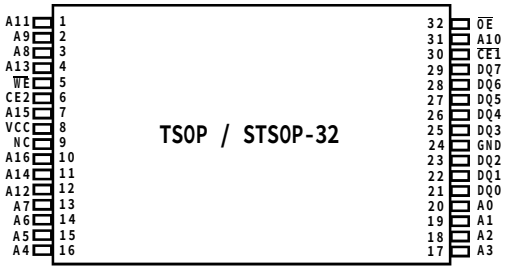
附录 G: STC 高性能 SRAM 选型一览表

型号	容量	工作电压	温度	速度	推荐封状	订货
STC62WV256	32K x 8	2.4 - 5.5V	-40℃~85℃	70nS	SOP/TSOP	订货
STC62WV1024	128K x 8	2.4 - 5.5V	-40℃~85℃	70nS	SOP/STSOP/TSOP	订货
STC62WV2568	256K x 8	2.4 - 5.5V	-40℃~85℃	70nS	STSOP-32	订货
STC62WV5128	512K x 8	2.4 - 5.5V	-40℃~85℃	70nS	STSOP/SOP-32	订货
STC62WV1M8	1M x 8	2.4 - 5.5V	-40℃~85℃	70nS	TSOP2-44	订货
STC62WV6416	64K x 8	2.4 - 5.5V	-40℃~85℃	70nS	TSOP2-44	订货
STC62WV12816	128K x 16	2.4 - 5.5V	-40℃~85℃	70nS	TSOP2-44	订货
STC62LV12816	128K x 16	2.4 - 3.6V	-40℃~85℃	70nS	TSOP2-44	订货
STC62WV25616	256K x 16	2.4 - 5.5V	-40℃~85℃	70nS	TSOP2-44	订货
STC62WV51216	512K x 16	2.4 - 5.5V	-40℃~85℃	70nS	TSOP2-44	订货

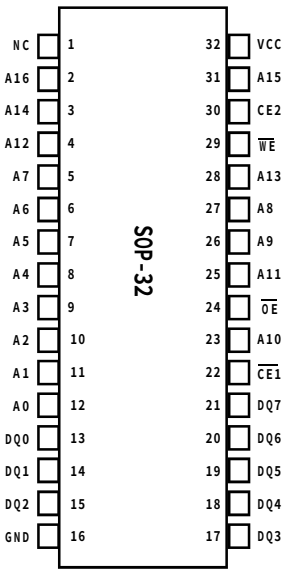
STC62WV256



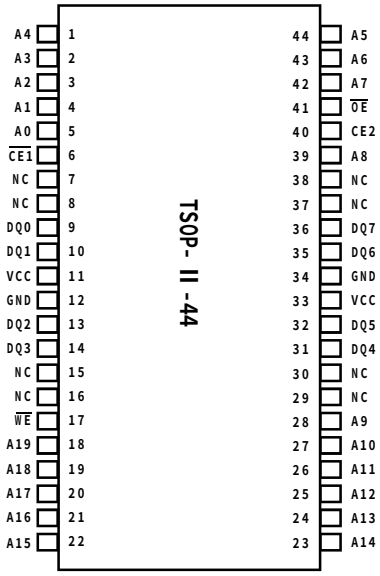
STC62WV1024



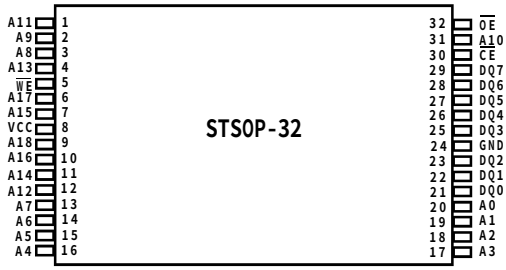
STC62WV1024



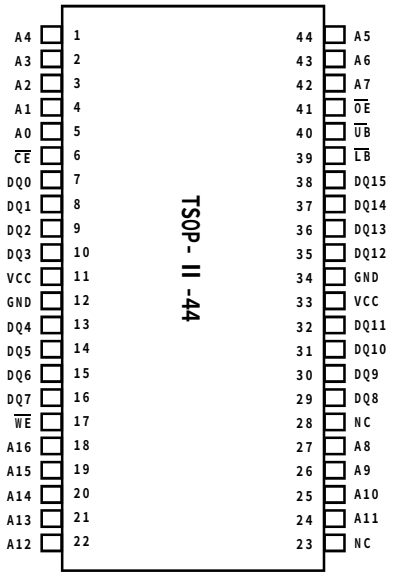
STC62WV1M8



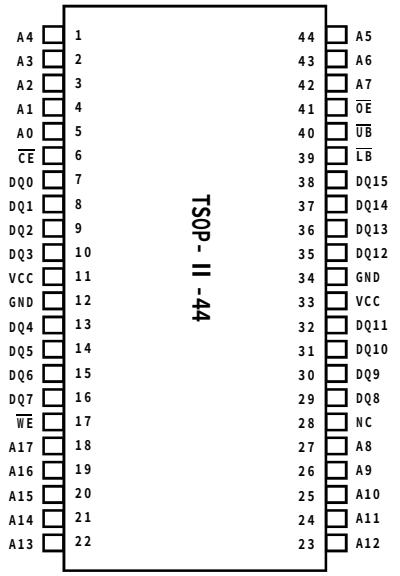
STC62WV5128



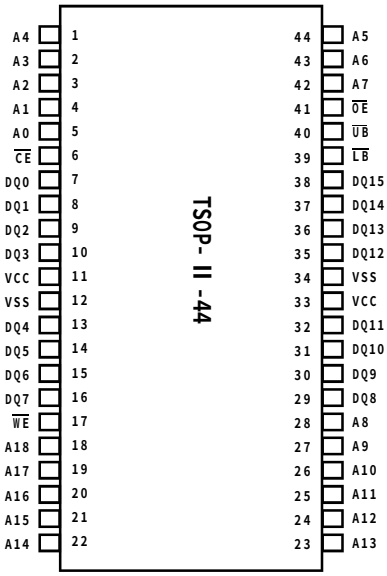
STC62WV12816



STC62WV25616



STC62WV51216



附录H： 超强抗干扰测试数据,过4000V快速脉冲干扰

--- 在权威的SGS电气实验室测试

Iss Date Jan 06, 2006

SGS

WORKSHEET： EFT Test (EN 61000-4-4)

EUT NAME：STC12C5410AD Development Tool

Environmental conditions

Air Temperature (℃)	20	Relative Humidity (%RH)	56	Air Pressure (mBar)	1003

Set-Up EUT for Compliance Test

Test Requirement Standard	EN 61000-4-4	Set up the EUT as Table-top
Operating mode during test	ON MODE	
Detail EUT monitor system: STC12C5410AD control LED flashing function		

Compliance-Test Record

Compliance Test Result On AC Cable (2 wire)			
Lead under test	Level	EUT operating mode	Observations
L	+/-4kV	ON MODE	NLOF
N	+/-4kV	ON MODE	NLOF
L+N	+/-4kV	ON MODE	NLOF

Test Conclusion and Reason

Test Conclusion: Pass

Reason(explain why?): Class A

Note: This sample pass the Class A(2kV)

According to the customer's require up to 4kV,test result: PASS 4kV

我们提供过4000V快速脉冲干扰辅导服务，陪同在权威的SGS电气实验室测试，1小时收取人民币3000元，提前一周预约。

附录 I： STC12C5410AD 系列单片机应用注意事项

关于复位电路：

晶振频率在 12M 以下时:可以不用外部复位电路，原复位电路可以保留，也可以不用，不用时复位脚可经过1K电阻短接到地,或者直接短接到地。不过建议设计时PCB板上保留R/C复位电路,实际使用时再决定用或不用。

关于时钟：

如果使用内部 R/C 振荡器时钟(4MHz～8MHz,制造误差加温漂),XTAL1 和 XTAL2 脚浮空.

如果外部时钟频率在27MHz以上时,建议采用实际基本频率就是标称频率的晶体,不要采用三泛音的晶体(基本频率是标称频率的1/3),因为外围参数搭配不当,时钟往往振荡在标称频率的1/3,即基频.或直接使用外部有源石英晶体振荡器,时钟从 XTAL1 脚输入,XTAL2 脚必须浮空.

关于 I/O 口：

少数用户反映 I/O口有损坏现象,后发现是

有些是I/O口由低变高读外部状态时,读不对,实际没有损坏,软件处理一下即可

是因为1T的8051单片机速度太快了,软件执行由低变高指令后立即读外部状态,此时由于实际输出还没有变高,就有可能读不对,正确的方法是在软件设置由低变高后加1到2个空操作指令延时,再读就对了.

有些实际没有损坏,加上拉电阻就OK了

是因为外围接的是 SPI/I2C 等漏极开漏的电路,要加 10K 上拉电阻.

有些是外围接的是NPN三极管,没有加上拉电阻,其实基极串多大电阻,I/O口就应该上拉多大的电阻,或者将该 I/O 口设置为强推挽输出.

有些确实是损坏了,原因:

发现有些是驱动 LED 发光二极管没有加限流电阻,建议加 1K 以上的限流电阻,至少也要加 470 欧姆以上

发现有些是做行列矩阵按键扫描电路时,实际工作时没有加限流电阻,实际工作时可能出现2个I/O口均输出为低,并且在按键按下时,短接在一起,我们知道一个CMOS电路的2个输出脚不应该直接短接在一起,按键扫描电路中,此时一个口为了读另外一个口的状态,必须先置高才能读另外一个口的状态,而8051单片机的弱上拉口在由 0 变为 1 时,会有 2 个时钟的强推挽高输出电流,输出到另外一个输出为低的 I/O 口,就有可能造成I/O口损坏.建议在其中的一侧加1K限流电阻,或者在软件处理上,不要出现按键两端的I/O口同时为低.

关于电源：

在电源两端应该加一个 10uF 以上的电解电容和一个 0.1uF 的小电容,进行电源去藕滤波.

附录J： 资料升级历史备忘录

※ STC12C5410AD 及 STC12C2052AD 系列单片机新的 C 版本已开始大量供货

2008-8-15 版本在 2006-8-7 版本的基础上:

增强了 STC12C5410AD 系列单片机电源管理寄存器 PCON 的应用说明，增加了 EEPROM 的使用注意事项。

2006-8-7 版本在 2006-6-26 版本的基础上:

STC12C5410AD 系列单片机和 STC12C2052AD 系列单片机几乎雷同，现将用户手册合为一体。

对 EEPROM 部分内容做了加强；对看门狗部分内容做了加强；对 I/O 口部分做了加强；对 PWM 部分做了加强；对掉电模式外部中断唤醒 C 语言程序做了加强；增加了 C 语言“ 自定义下载演示程序 ”，可以实现不停电下载；增加了如何让 I/O 口上电复位时为低电平

2006-6-26 版本在 2006-6-10 版本的基础上：

部分目录和内容做了调整，并增加了用外部中断唤醒掉电模式的 C 语言测试程序

2006-6-10 版本在 2006-4-30 版本的基础上：

部分目录和内容做了调整

2006-4-30 版本在 2006-4-15 版本的基础上：

部分目录和内容做了调整

2006-4-15 版本在 2006-3-25 版本的基础上：

部分目录和内容做了调整

通知新版本 C 版本已经大量供货

2006-3-25 版本在 2006-2-6 版本的基础上：

在 PWM/PCA 的应用部分增加了在使用 PCA 高速输出模式时的特别应用注意事项

2006-2-6 版本在 2005-1-16 版本的基础上：

增加了定时器1做波特率发生器的程序

增加了如何用软件实现系统复位

增加了附录： 典型 MCU/DSP/uC 复位、电源监控、外部看门狗专用电路

增加了附录：STC 高性能 SRAM 选型一览表

增加了附录：超强抗干扰测试数据,过4000V快速脉冲干扰

对 EEPROM 测试程序的解释说明部分做了加强

对看门狗测试程序的解释说明部分做了加强

2006-1-16 版本在 2005-12-31 版本的基础上：

1.A/D 转换程序做了简化

2.PCA/PWM模块增加了新的演示程序(扩展软件定时器,扩展外部中断)

2005-12-31 版本在 2005-12-24 版本的基础上：

1.修正了 PCA/PWM 部分笔误，4 路 PCA/PWM 原有些部分笔误为 2 路

2.原 A/D 转换结果计算公式 ：笔误 结果(ADC_DATA[7:0],ADC_LOW2[1:0])=256 x Vin / Vcc 现改为 结果(ADC_DATA[7:0],ADC_LOW2[1:0]) = 1024 x Vin / Vcc

3.增加了 STC 单片机普通 I/O 口驱动 LCD 显示的参考电路及演示程序

4.增加了一个 I/O 口驱动发光二极管并扫描按键

5.增加了 STC12C5410 系列单片机应用注意事项